UNITED STATES AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

1

AFIT/GCS/EE/81D-10

Design And Implementation Of USAF Avionics Integration Support Facilities

Thesis

AFIT/GCS/EE/81D-10 Jon G. Hanson
Captain USAF

Approved for public release; distribution unlimited

AFIT/GCS/EE/81D-10

DESIGN AND IMPLEMENTATION OF USAF AVIONICS INTEGRATION SUPPORT FACILITIES

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Jon G. Hanson, B.S.
Captain USAF
Graduate Computer Science

Approved for public release; distribution unlimited

Preface

This work presents an SADT Actigram model of the functional requirements for implementing an Avionics Integration Support Facility. The functional requirements are universal among avionics systems though, until standardization of avionics systems/subsystems is complete, implementations will remain system-specific. Also presented here is a Q-GERT simulation of a specific implementation of a portion of the functional requirements.

SADT is a technique developed by the SofTech Corporation, of Waltham, Massachusettes. Research accomplished by the TRW Defense and Space Systems Group also proved quite helpful in accomplishing this investigation.

I would like to express my deep appreciation to Dr. Gary B. Lamont, my research advisor who provided valuable guidance and encouragement. I thank my thesis readers, Mr. Ray Rubey of the SofTech Corporation, and Professor Charles Richard, whose constructive comments helped me to improve the quality of this work. I also thank Captain Steven Cristiani, of SEAFAC, the organization who sponsored this investigation, for his guidance and advice in my efforts.

Finally, I wish to thank my wife, Linda, for her limitless love and support during the past year, and the many hours she spent helping me with the graphics presented here.

Jon G. Hanson

Contents

Page

Page

List of Figures

Figure Page

Figure Page

List Of Tables

Abstract

The design and implementation of avionic software support facilities known as Avionic Integration Support Facilities (AISFs) are investigated. A complete set of AISF functional requirements is presented in Data Flow Diagram format. This method of presentation facilitated the hierarchical development of the functional requirements. From these functional requirements an implementation model for such a facility is developed. The modeling vehicle used is Softech's Structured Analysis and Design Technique (SADT). The SADT model depicts inputs, processes, outputs, controls, and the mechanisms required to implement the AISF.

Following completion of the SADT model a specific implementation of a portion of an AISF, the Integration Test Bed (ITB), is examined. The ITB considered is an Avionics Modernization Hot Bench presently being developed by the Aeronautical Systems Division, System Engineering Avionics Facility (SEAFAC). A portion of the hot bench, the real time simulation software, is modeled and analyzed with the Queuing Graphical Evaluation and Review Technique (Q-GERT). Q-GERT is a modeling vehicle and computer analysis tool well suited for this purpose. In this way a predictive model of the real time simulation software is provided and the viability of the hot bench design verified.

A brief investigation of possible future AISF configurations is presented along with a discussion of presert standardization efforts in the avionic arena and their effects on avionic software support.

I. Introduction

Over the past decade, there has been an increasing number of Embedded Computer Systems (ECS) integrated into Air Force weapon systems. (Ref 23:126) The ECS, along with its software the Operational Flight Program (OFP), implements or aids in the implementation of system and subsystem performance, and integrates the system elements into a highly effective weapon system. The effective change and engineering support of ECS software requires specialized facilities, skills, and equipment. The Avionics Integration Support Facilities (AISF), located at the Air Logistics Centers (ALC), provide a laboratory environment where OFPs are reprogrammed and ECS engineering is performed including: investigative engineering, system performance evaluation, analysis and resolution of hardware and software problems, and the integration and testing of the ECS.

The purpose of this investigation is to establish a standard software development system applicable through the life cycle of an ECS and applicable to the AISF. A model of the software facilities required by the AISF will be presented.

Historical Perspective

Embedded Computer Systems have contributed to the realization of Air Force systems with improved performance, capability, and flexibility. (Ref 23:116) They are programmable devices exhibiting high speed, accuracy , and reliability in processing and manipulating data, performing computations and exercising control.

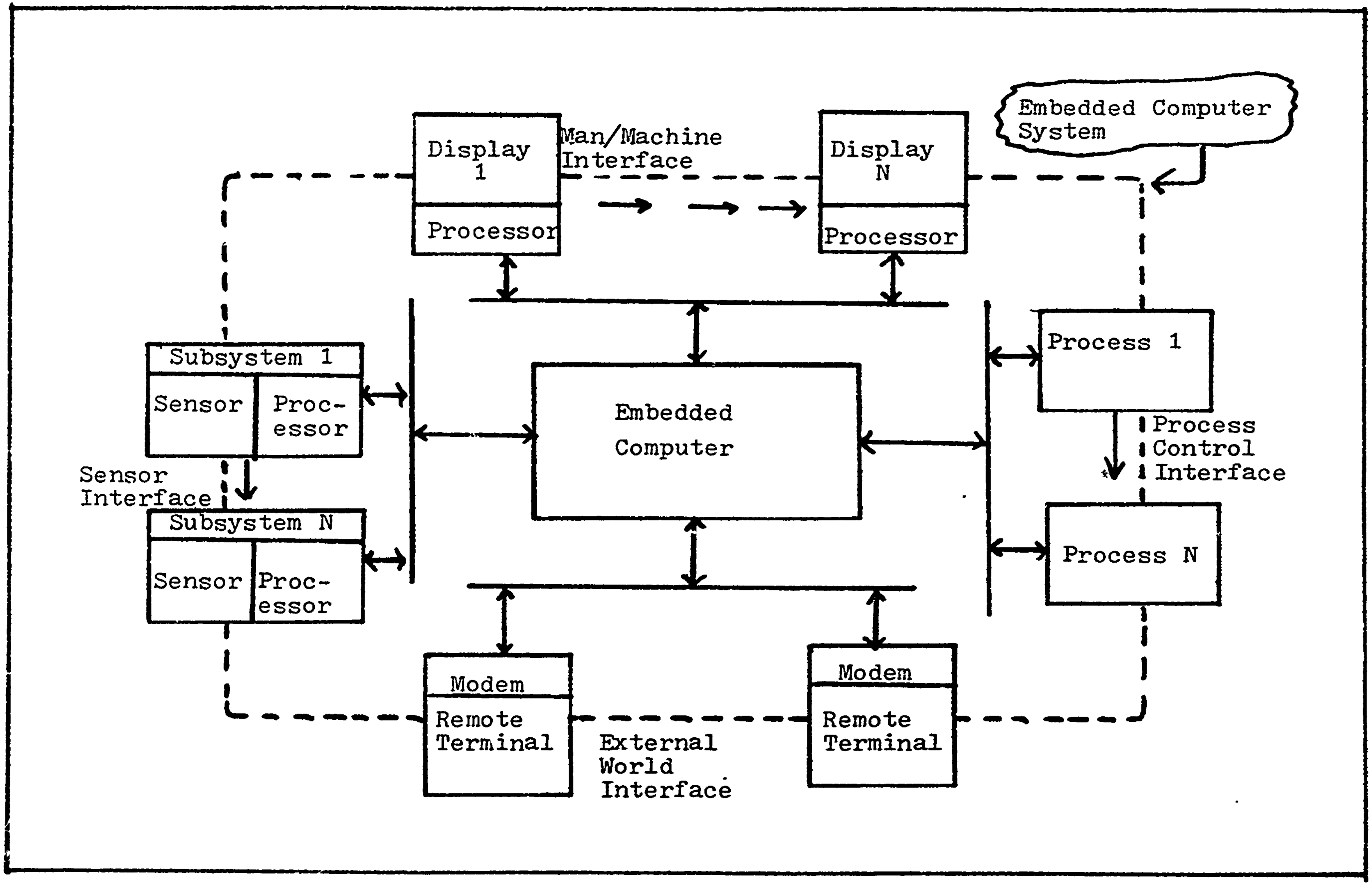


Figure 1. Typical Weapon System (Ref 23:1)

An ECS may be configured as the central element of an aircraft navigation and control system. In this role, the ECS receives sensor, mission, and aircrew inputs. Operating on these inputs, it then computes navigation, steering, and weapon delivery functions, and outputs data and commands to the cockpit displays, flight controls, and stores management systems. These outputs are used by the aircrew in determining tactics. Figure 1 depicts a typical weapon system with an ECS.

The reprogrammable aspect of the ECS potentially provides the weapon system with great flexibility. In most uses the embedded computer executes software programs (OFPs). Weapon system performance can be more easily modified or enhanced by changing the OFP than by hardware change. The reprogramming of the computer can be accomplished much faster and much more economically than the development of new hardware. This feature of the ECS provides the potential for economically maintaining the weapon system's performance in accordance with changing user requirements over the operational life of the weapon system. (Ref 23:126)

The development of an OFP for a weapon system begins in the analysis phase of its life cycle and is carried through the design, coding and checkout, test and integration, and the operational phase, the longest in the life cycle, will last from ten to thirty years depending upon the weapon system. During this phase, OFPs require software changes every one to four years. (Ref 24:5) These engineering and reprogramming tasks are accomplished at the Air Force Logistics Command's Avionics Integration Support Facilities (AISFs), located at each Air Logistics Center

(ALC). (Ref 23:126)

The Integration Support Facilities (ISFs) provide a myriad of services which contribute to the solution of the user's day to day problems. The capabilities of the ISF are summarized in Table 1. The size and sophistication of these facilities are highly dependent upon the size and number of ECSs being supported. They are in most cases colocated with the system and are staffed with Air Force Personnel. (Ref 23: 127)

TABLE 1

Integration Support Facility Capabilities

-Addition and/or change in systems capability.

-Deletion, addition and/or change in operational modes.

-Changes in operational functions.

-Reliability and/or maintainability improvements.

-Changes and/or replacement of system equipment.

-Improvement of test capabilities.

-Correction of system hardware and software deficiencies.

-Corrections to operations and/or operational procedures.

-Corrections to equipment problems.

-Compensation for equipment degradation.

-New Developments.

This approach is intended to centralize essential management and engineering functions for the system, establish a close working relationship between management, systems and ECS engineering, and enable the ISF to more effectively deal with the idiosyncracies of a particular system. The advantages of this approach, greater Air Force control, coordinated support activity, and continuous uninterrupted support, generally result in more responsive support. (Ref 23:127)

ISF equipment is composed of both laboratory and weapon system resources. The ECS support requirement determines the quantity and sophistication of the equipment, but as a minimum, the ISF generally includes a host computer, dynamic simulator, and an ECS mock-up. The computer system and some features of The dynamic simulator have capabilities enabling them to be shared among all ECSs supported by the ISF, however the ECS mock-up is generally peculiar to a particular system, and may not be a complete replication of the weapon system ECS. The principal tool used in the ISF to test and accomplish reprogramming changes for OFPs is the dynamic simulator. These simulators provide a real world dynamic environment for quantitative analysis and test and evaluation of ECS software changes. They are used for investigating ECS system and software problems. The simulators, configured in multi-mode, provide the capability to monitor ECS performance in real time. The various modes include man-in-the-loop, canned scenario, and non-real time operation. (Ref 23:128) The man-in-the-loop mode provides the capability for evaluating man-machine interfaces and real time operation. The canned scenario mode enables the performance of trend and sensitivity analysis and provides a controlled environment. The non-real time mode provides the capability to load and verify OFPs. When coupled with the host computer system, the dynamic simulators can also provide support to the initial code and debug effort in software development. Use of the dynamic simulator for the development of OFP changes results in a high degree of confidence that the software is correct when released for operational use. (Ref 23:128)

Test and evaluation of interfaces between the ECS computer and other subsystems is accomplished on the ECS Integration Test Bed (ITB), mock-up, or, hot test bench. The ˙ ɔt bench functionally integrates the actual ECS hardware with the software and provides a capability to test timing and hardware/software compatibility. The hot benches provide the capability for analyzing ECS system problems at the system, subsystem, and module level. Among the problems which can be successfully diagnosed using a hot bench are those which are application-dependent, time-dependent, and intermittent. Hot benches also provide an excellent environment for duplicating user problems and evaluating solutions.

The host computer provides the ISF staff with a general purpose interactive computing capability, which allows them to code and debug programs, cross-compile, and cross-assemble OFPs and source data. The host computer also provides the ability to generate code and documentation, store format, manipulate data, maintain engineering and management data, and reduce/analyze test data. (Ref 23:129)

The Air Force faces a rapidly expanding critical mission need to support new requirements for digital weapon systems. With the projected influx of ECSs into the Air Force inventory a heavy demand will be made upon personnel and other resources required to support these systems. (Ref 11:2) Applying current methods of ECS support to this requirement could result in a shortage of personnel and funds This deficit along with the accelerated use of OFPs in weapon systems will result in a decreasing AFLC support posture as additional ECSs are introduced into the inventory.

For these reasons an integrated, automated, modular, skills-hierarchical approach to the software support capability must be developed to insure timely support to the combat mission from system engineering development through the life of the system. Standardized, modular, and automated hardware and software support tools are needed at each ALC to improve current support. ECS hardware and software standardization must be the norm. (Ref 11:2)

Objective

In this investigation a complete design for an Avionics Integration Support Facility (AISF) will be developed. Though hardware, facilities, personnel, and software requirements will be considered, the emphasis will be on software. Software requirements in the form of development tools and techniques will be considered. Hardware will mainly be considered as it interfaces with the required software.

Based upon the complete design for an AISF a model will be presented and implemented using the Structured Analysis and Design Technique (SADT) design tool. SADT lends itself especially well to this purpose because it provides a methodology for performing functional analysis and design. It provides the techniques for thinking in a structured way about large and complex problems and communicates the results in a clear, concise notation. A portion of the model, the Integration Test Bed (ITB) will then be simulated using the Queuing Graphics Evaluation Review Technique (Q-GERT) modeling and analysis language. Q-GERT is an automated simulation tool designed for studying the procedural aspects of systems. The design of Q-GERT is hierarchical allowing the modeler to extend or revise the model. The ITB to be simulated is an Avionics Modernization Hot Bench (AMHB) being developed by the Aeronautical Systems Division (ASD) of the USAF Systems Command.

Approach

The ultimate goals of the investigation are to reduce the proliferation of AISFs in the Air Force, to save the Air Force money with a standard AISF design, and, to present a complete SADT model of the software requirements of the AISF. It is also the goal of this investigation to provide to the Systems Engineering Avionics Facility (SEAFAC) Branch of ASD, a predictive simulation model of their AMHB.

These goals were accomplished in three phases. Phase One involved a thorough literature search in order to determine the functional and operational needs of such a facility. Though research included all aspects of the AISF, concentration was focused on software facilities. Research was accomplished in the area of facility standardization and how it will apply to the functional and operational AISF. Present moves toward standardization such as the MIL-STD 1553B Multiplex Data Bus, MIL-STD 1750A Computer Instruction Set Architecture, and the

MIL-STD High Order Language (HOL) Jovial and future HOL Ada, and their impact upon the job of avionics software development and maintenance were explored. Research on the impact of standardization upon such factors as cost, schedule, and performance of the AISF and Embedded Computer Systems (ECS) was conducted.

In Phase Two research resulted in determination of software facilities required for an AISF. This phase produced the SADT model of the AISF. In developing the AISF model a top-down development approach was used, allowing the design to first address those levels closest to the users, and to insure that the design was consistent with user requirements of the next higher level. This top-down approach, coupled with the use of SADT, yielded clear written specifications and diagrams that will provide to those implementing the design the required unambiguous information.

In Phase Three, a portion of the model developed in the previous phase was examined and simulated using the Q-GERT language. A predictive model of the real time simulation portion of the KC-135 AMHB was developed and run in order to examine functional characteristics of the AMHB.

Overview of The Thesis

The structure of the thesis goes from high-level to detailed to high-level. Chapter II provides a brief treatment of the process of deriving functional requirements. In Chapter III the functional requirements for the complete AISF are presented in Data Flow Diagram format. The next chapter provides the more detailed SADT model of those functional requirements including controls and mechanisms affecting and facilitating the functions. Contained in Chapter IV are only top-level SADT diagrams for all but the Integration Test Bed portion of the model. The complete model is contained in Appendix C. The detailed model for the ITB portion of the AISF is provided in Chapter IV because the next chapters deal specifically with it.

In Chapter V a detailed discussion of the design of an ITB being implemented at SEAFAC is presented. The ITB is discussed with respect to the functional requirements of Chapter III and the model of Chapter IV. In Chapter VI the ITB's real time simulation software is modeled and analyzed with Q-GERT. Five experiments were conducted to determine if SEAFAC's design is viable and conclusions and results of these experiments are presented. Supporting data are located in Appendix E.

Chapter VII once again deals with the total AISF. Present moves towards avionic systems standardization are discussed along with their possible impacts on the AISF concept. Two new concepts for standard AISFs of the future are presented and discussed.

Finally, Chapter VIII summarizes this investigation and provides recommendations for further research.

II. Determining Functional Requirements For Development Specifications

Introduction

This chapter describes and illustrates the procedure of deriving the functional requirements to be used in producing a design for a standard Avionics Integration Support Facility (AISF). Though requirements for hardware, software, facilities, and personnel must be considered, the emphasis here will be upon determination of software requirements. First, a general discussion of, and justification for, functional requirements is given. Next, the requirements analysis procedure is discussed and accompanied by a set of SADT diagrams illustrating the process. Finally, functional requirements are discussed and the procedure for determining them is presented in SADT format. The chapter concludes with a brief discussion of performance requirements, and development standards and constraints.

Background

A complete and accurate set of software functional requirements is fundamental to the development of a standard AISF. The purpose of establishing requirements for the AISF is to specify to those who will implement the facility all the known requirements of the software which will be required for the processes to be accomplished by the AISF. Definitive software requirements are essential to ensure AISF software which will accomplish the mission objectives and provide for expansion to accomplish future objectives while minimizing the cost of the AISF. The software requirements derived from system specifications establish functional, performance, and interface requirements that the AISF software must be designed to meet. They establish development standards and constraints to which the AISF software and its documentation must comply. Good software requirements will convey to those actually implementing the facility precisely what is required for mission accomplishment. This will enable the Air Force Logistics Command (AFLC) to define a viable technical approach and arrive at an accurate estimate of the AISF software development cost. It also increases the probability that the end product will satisfy Air Force mission objectives. Good requirements produced early in the facility development process will pay off economically in terms of dollars and better AISF performance as well as facilitating top-down design, meaningful testing, and management visibility and control. (Ref 6:3)

Software Requirements Analysis

A prerequisite for specifying software requirements is a software requirements analysis. A thorough analysis of objectives and interfaces is necessary to derive and document specific and quantitative AISF software requirements. The AISF software functional specification should include all requirements with which a standard facility will have to comply to perform its intended function adequately. The AISF software requirements describe the functions that the AISF software must perform, the environment in which the

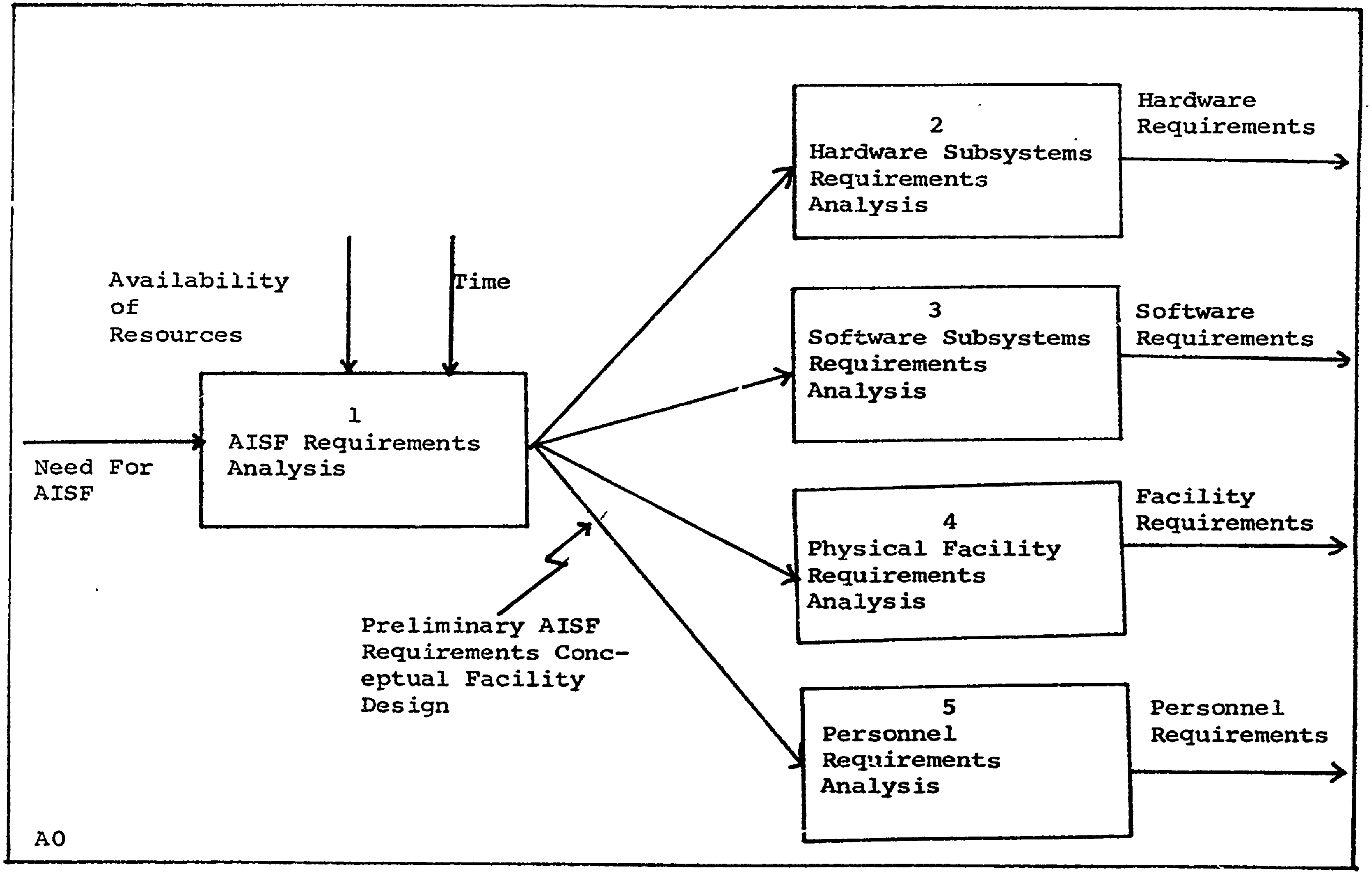


Figure 2. Systems Engineering Process

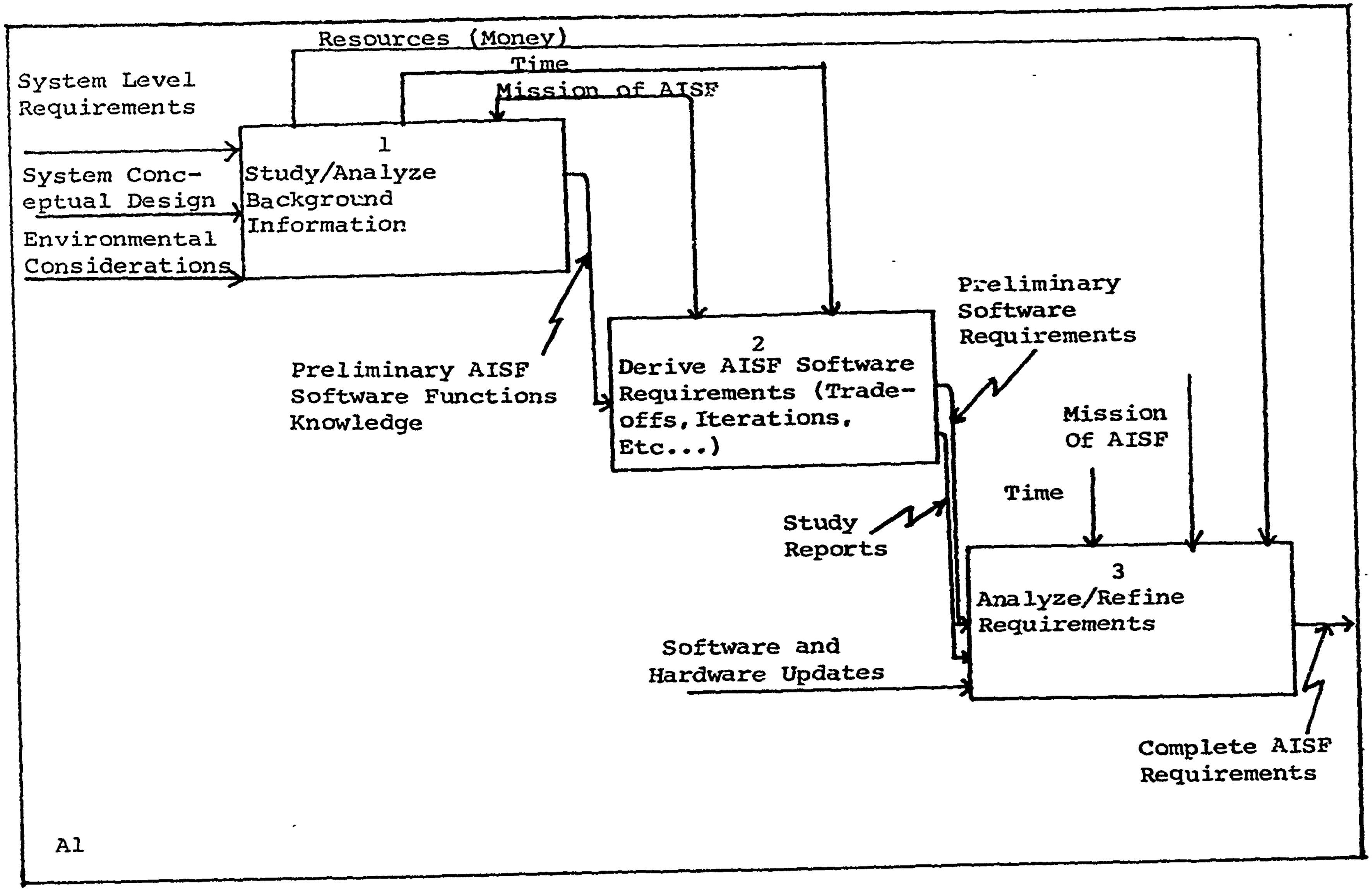

Resources (Money)
Time
Mission of AISF
System Level Requirements
System Conceptual Design
Environmental Considerations
1
Study/Analyze Background Information
Preliminary AISF Software Functions Knowledge
2
Derive AISF Software Requirements (Trade-offs, Iterations, Etc...)
Preliminary Software Requirements
Study Reports
Software and Hardware Updates
Mission Of AISF
Time
3
Analyze/Refine Requirements
Complete AISF Requirements
A1

Figure 3. AISF Requirements Analysis

software must operate and constraints and standards that must be satisfied during AISF software design and development. Figure 2 depicts the system engineering process that results in requirements that will define the AISF. The process involves a hierarchy of requirements generation, beginning with an operational mission need and ending with detailed engineering specifications and data. Each level of requirements, beginning with functional requirements, leads to the next lower level of detail until the entire facility is specified. The process continues to define and optimize the requirements on subsystems, comprising the entire AISF. This engineering data is then analyzed to determine design requirements. Engineering and tradecff studies are performed to determine the best approach to a total integration facility configuration. This effort results in a system specification for the AISF. (Ref 6:1-5)

The Requirements Analysis Procedure

Software requirements analysis is an interactive process which may require numerous iterations of tradeoffs to determine the best possible approach. Life cycle costs are also part of the selection criteria for determining not only the types of computer equipment and programs, but also the subsystem architecture and allocation of requirements to subsystem elements.

Figures 3 through 5 depict the AISF software requirements analysis procedure. Figure 3 gives an overall view of the process of requirements generation. This process

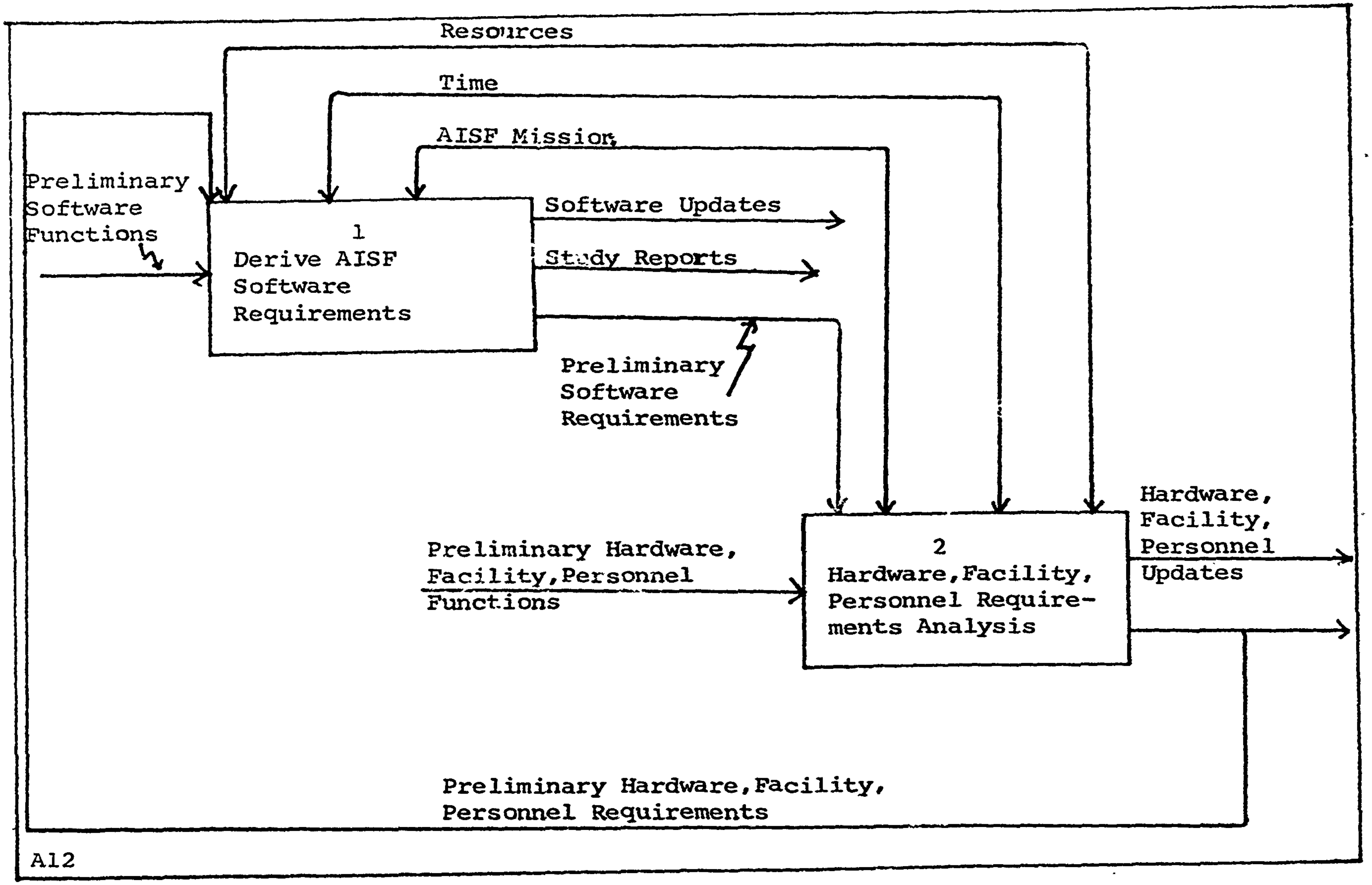

Resources
Time
AISF Mission
Preliminary Software Functions
1 Derive AISF Software Requirements
Software Updates
Study Reports
Preliminary Software Requirements
Preliminary Hardware, Facility, Personnel Functions
2 Hardware, Facility, Personnel Requirements Analysis
Hardware, Facility, Personnel Updates
Preliminary Hardware, Facility, Personnel Requirements
A12

Figure 4. Iterations and Tradeoffs in Requirements Analysis

occurs in three phases and results in the complete set of requirements for AISF software. Box 1 represents phase I which consists of studying the system level requirements, AISF conceptual design and the environment the AISF is expected to support. The first step is to study and thoroughly understand the AISF conceptual design and system level requirements and to determine their impact on the AISF software requirements. This analysis yields a preliminary set of AISF software functions.

The environment which the AISF must support and operate in is also studied during phase I. This includes considerations such as types of systems to be supported and extent of support to be provided by the AISF.

Phase II, which is shown in Figure 4, consists of deriving the detailed AISF software requirements, the functional and performance requirements, and development standards and constraints. In order to produce requirements which correspond to acceptable system performance, while minimizing AISF cost, numerous tradeoffs, iterations, and studies must be performed. Figure 4 shows hardware, facility, and personnel requirements analysis occurring simultaneously to the software requirements analysis. These two analyses must be interactive so that the software requirements are compatible with the other requirements and to minimize AISF cost. Hence the hardware/facility/personnel requirements analysis results must be made available to the software requirements analysis and vice versa. The net result of a successful phase II is a baseline set of AISF software requirements.

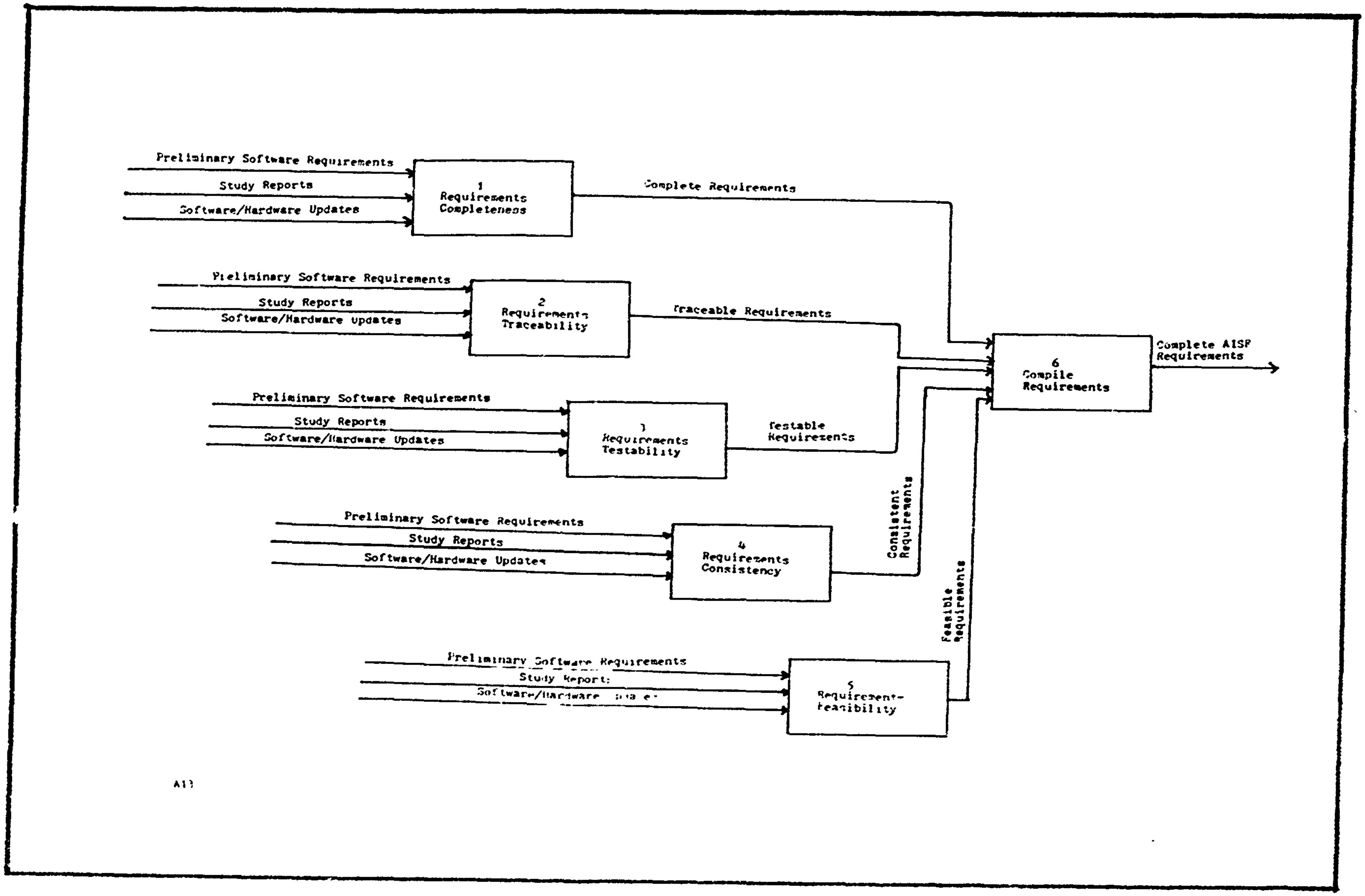


Figure 5. Analyze and Refine Requirements

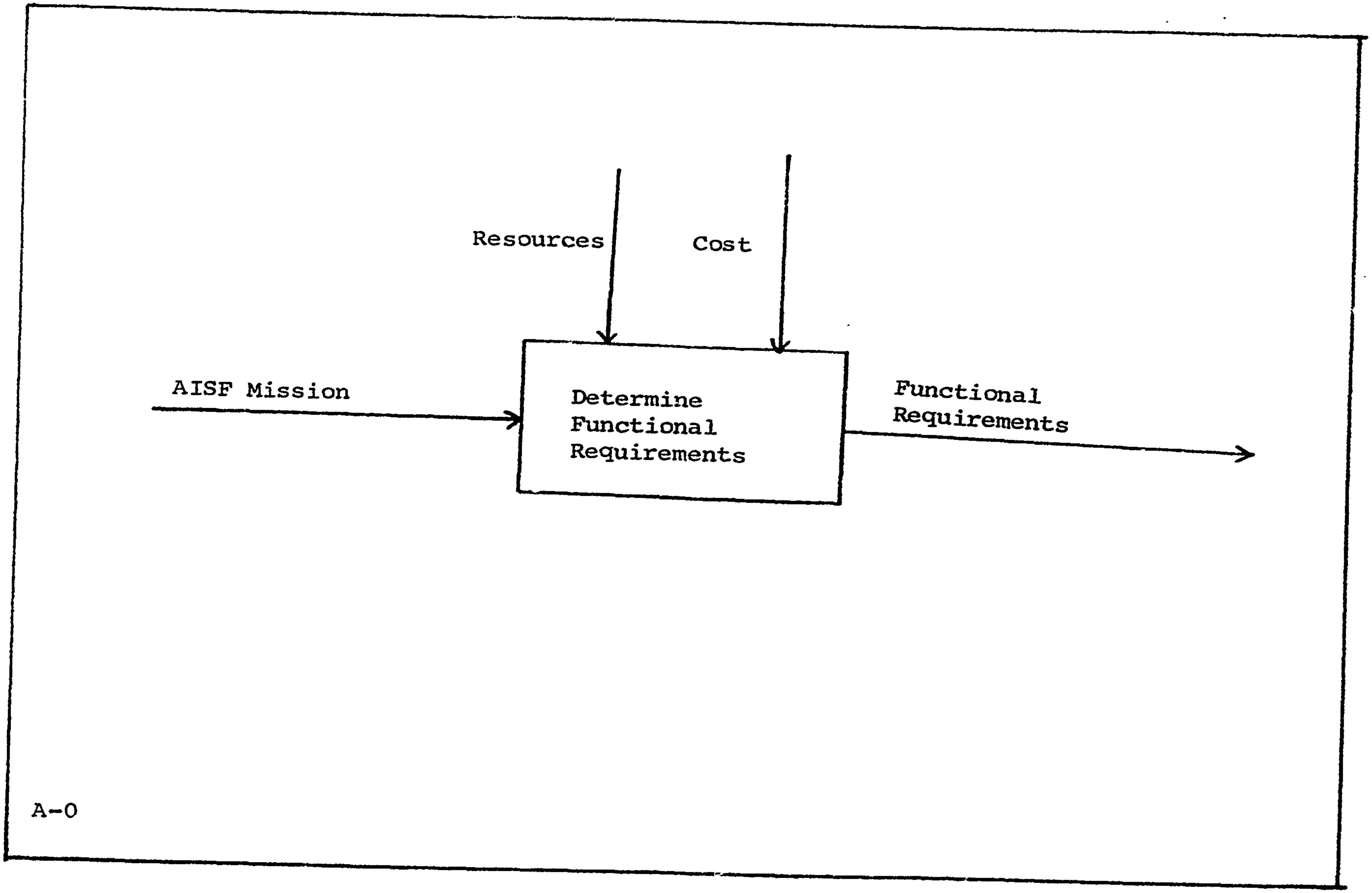


Figure 6. Determine Functional Requirements (Context)

Phase III of the software requirements analysis procedure is an assessment of the requirements determined in phase II relative to completeness, traceability, testability, consistency, and feasibility. This phase is illustrated in Figure 5. This phase also involves factoring in updated hardware/personnel/facility characteristics into the requirements. The third phase may result in many changes to the requirements produced by phase II. This is part of the iterative process that yields good software requirements.

Functional Requirements

Functional requirements for the AISF software specify the processing it performs. An AISF software function may include decision logic as well as arithmetic processing. "The functional requirements describe the inputs, outputs, and processing performed by each function and they specify the quantitative performance requirements of each function when applicable." (Ref 6:34)

In Figures 6-9 the process of deriving a complete set of functional requirements is illustrated. The process of deriving functional requirements begins with establishing and defining all the functions that must be performed by the AISF's software. Functions are identified during the requirements analysis as a result of studying the mission the facility is to perform and the type of hardware that the software must run on. The combination of all AISF software functions constitutes all desired processing capabilities of the software. Figure 7 depicts the process of deriving

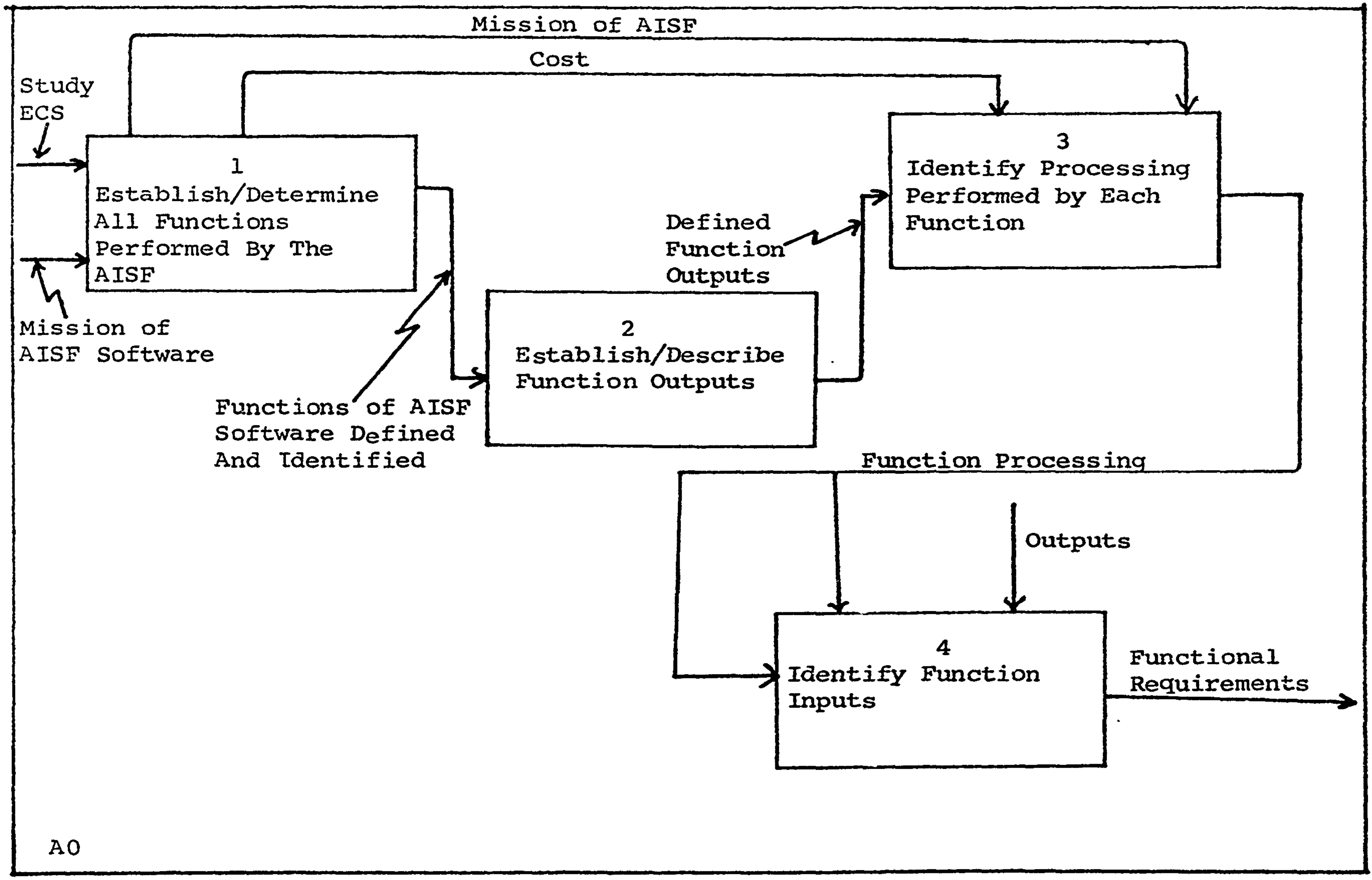


Figure 7. Determine Functicnal Requirements

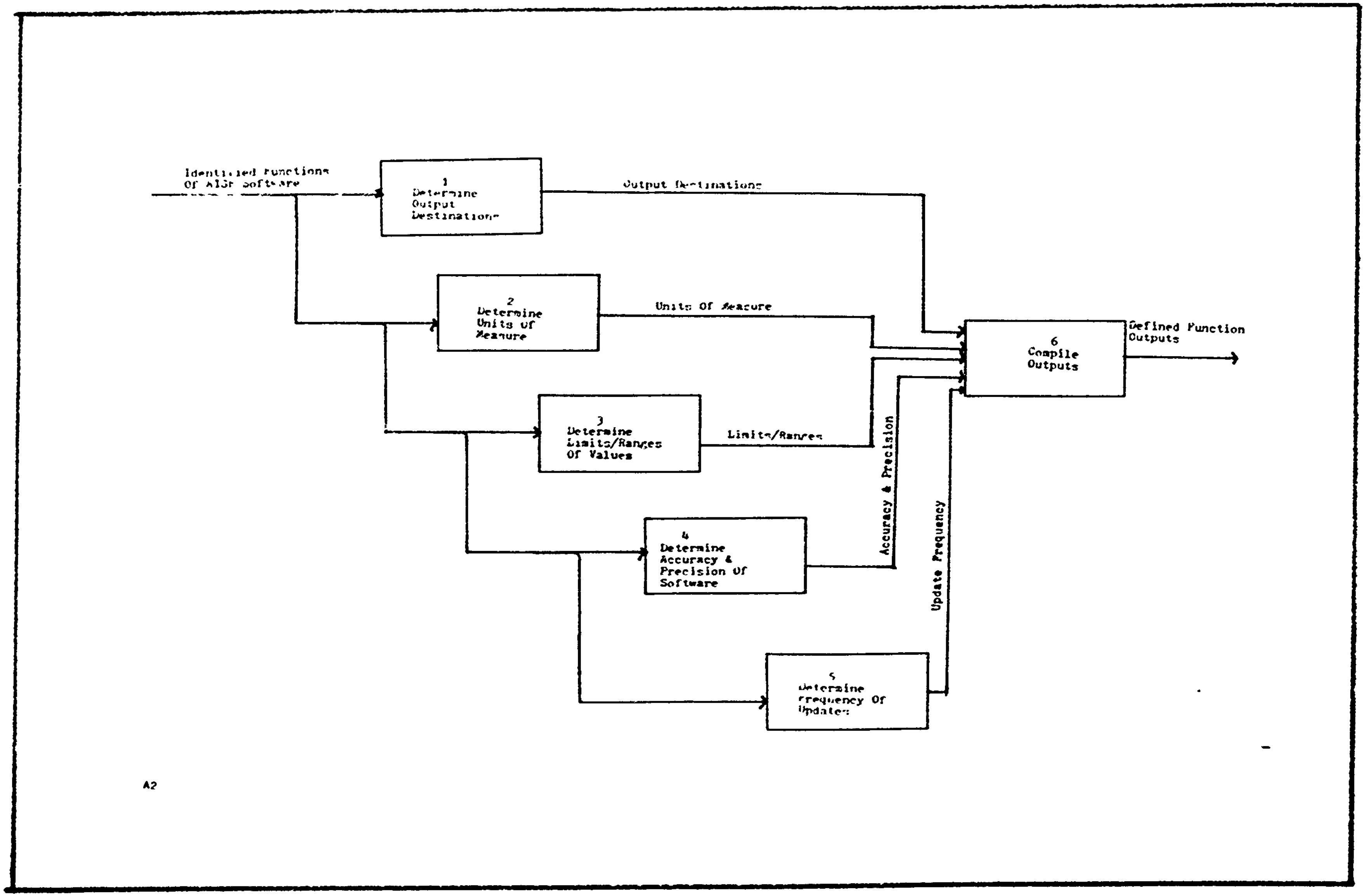

Identified Functions Of AISF Software
1 Determine Output Destinations
Output Destinations
2 Determine Units Of Measure
Units Of Measure
3 Determine Limits/Ranges Of Values
Limits/Ranges
4 Determine Accuracy & Precision Of Software
Accuracy & Precision
5 Determine Frequency Of Updates
Update Frequency
6 Compile Outputs
Defined Function Outputs
A2

Figure 8. Defining Function Outputs

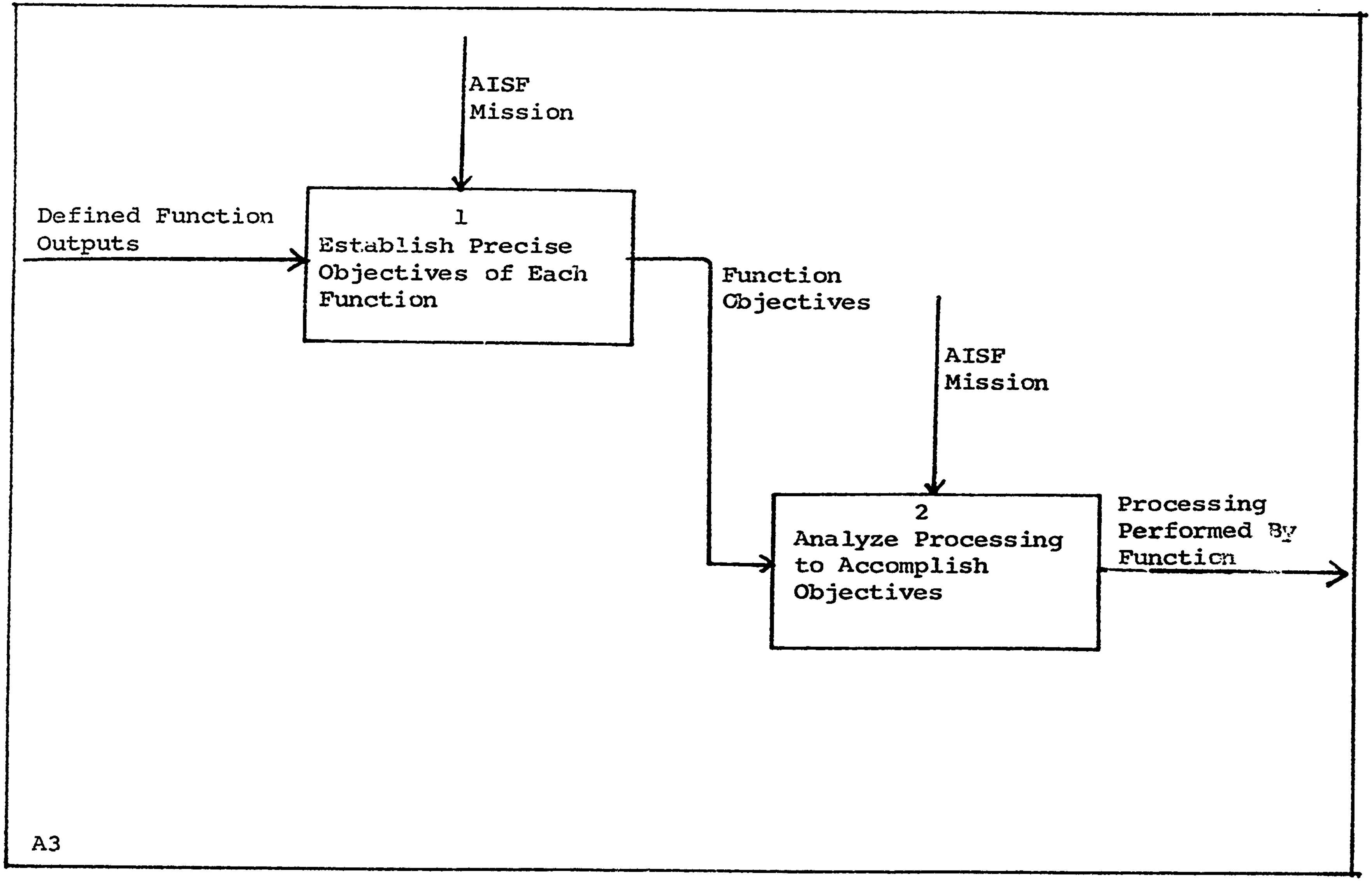


Figure 9. Establishing Processing Performed by Functions

functional requirements. Based upon the systems to be supported and the mission of the AISF, all functions to be performed by it are determined. Identified functions serve as inputs to a process for determining function outputs based upon the mission of the AISF. This process is further detailed in Figure 8. The input to the process is the set of identified functions of the AISF software. The process of determining and identifying function outputs involves determination of function aspects such as output destinations, units of measure, limits and ranges of values, accuracy and precision required, and frequency of update. Once the function outputs are established, the processing to be performed by each function must be identified. This process is detailed in Figure 9 and is accomplished by first establishing the precise objectives of each function and then analyzing what processing is required to accomplish those objectives.

After the processing requirements of a function are determined, the required inputs to the function can be identified and described, including the sources of the input data, units of measure, limits and ranges of values, precision, and frequency of the input data.

Performance Requirements

Performance requirements specify in quantitative terms how well the AISF software must satisfy its processing objectives. A performance requirement may be directed at a single function or it may be directed at the AISF software system as a whole. A performance requirement may be based on system level requirements imposed upon the AISF by other systems that interface with it. A performance requirement is generally used when approximations, compromises, and tradeoffs must be made in the design in order to satisfy a processing objective within system constraints such as memory or throughput limitations.

Development Standards and Constraints

The software development standards and constraints specify how the AISF development contractor is to accomplish the following during software development: assurance of compatibility among computer program components, ensure an orderly, systematic development procedure, ensure adequate software documentation and provide for future modification and growth of the software. Compliance with development standards and constraints should result in a well documented, high-quality software product and a reduction of management, check out and testing costs during software development.

Design Standards

Generally, software design standards require the use of a top-down procedure, however, when some software already exists, a combination of bottom-up and top-down design approaches may be used. Top-down design, which starts with the design of the top level function and proceeds downward in the design of successively lower level functions, enhances design traceability, completeness, and comprehensiveness.

Top-down design is initiated by establishing a functional hierarchy, with the top level function in the hierarchy, the overall mission to be performed by the software. Lower levels are obtained by breaking down the software functions into smaller functions of greater detail. The software functional requirements must be mapped onto the hierarchy.

After the functional hierarchy is established, the design of each function can occur, starting with the top level function and proceeding downward. Any discrepancies between the software functional requirements and design should be resolved at each level of design.

Summary

In this chapter the actual mechanics of performing a requirements analysis and deriving specifications for the AISF software were discussed. This approach is the one employed in this investigation to write and model, in SADT, a development specification for a standard AISF.

The production of a development specification for a standard AISF began with a requirements analysis. Though all aspects of the AISF were considered, including hardware, software, facility, and personnel, the emphasis was placed upon the software required by the AISF to accomplish its mission. The results of the requirements analysis and determination of functional requirements was a complete set of functional requirements for the standard AISF. These requirements were translated directly into the DFD model of Chapter III for SADT modeling. The SADT model comprises Chapter IV.

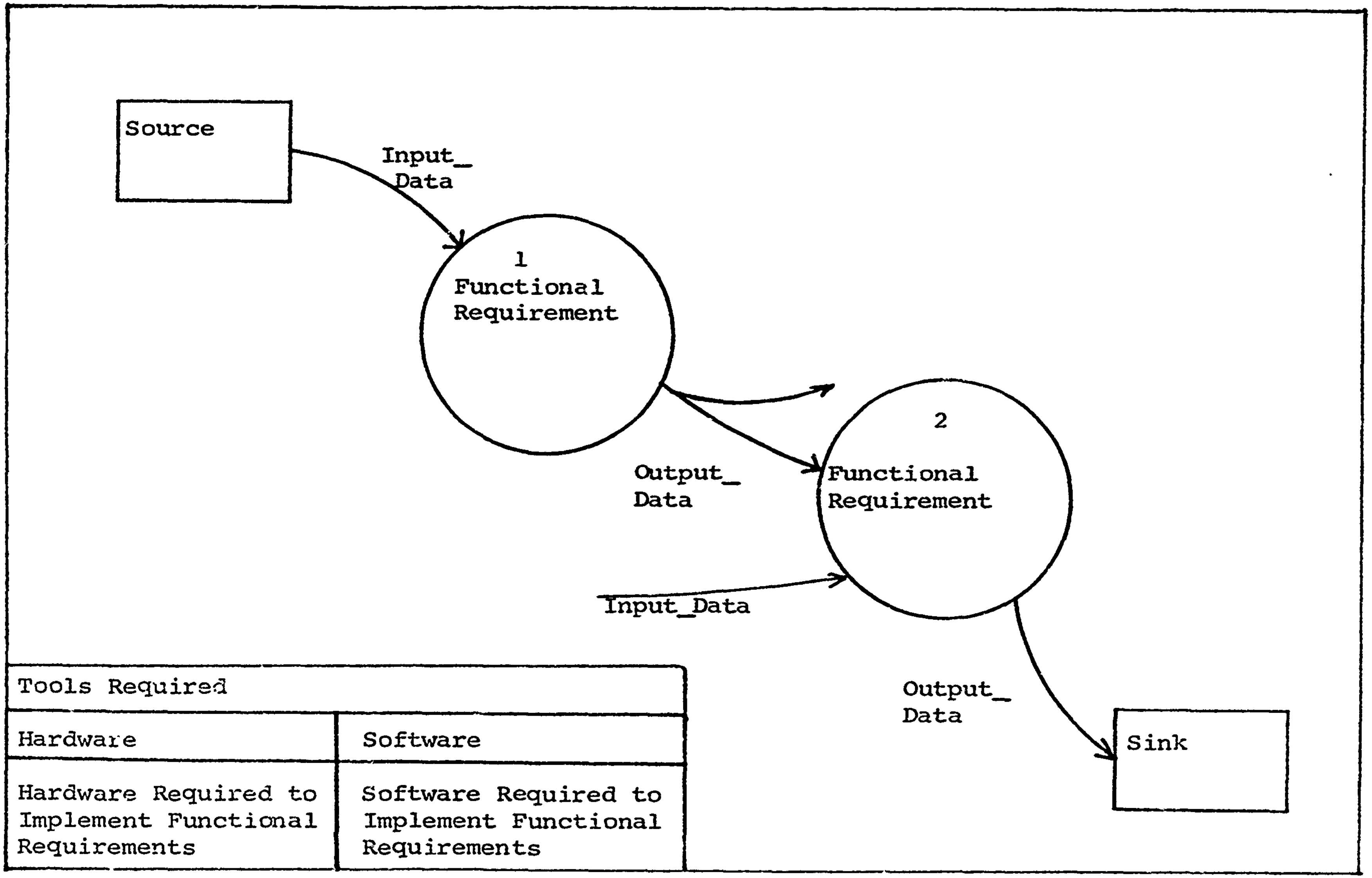
Design standards in the form of a top-down approach and structured design were imposed upon the development specification. Design and performance requirements, as well as development standards, were considered.

III. AISF Functional Requirements

Introduction

This chapter outlines the functional requirements for an AISF. The OFP change process accomplished at the AISF is similar to the Full Scale Engineering Development (FSED) process of producing software. For this reason the FSED process is discussed first in order to familiarize the reader with the basic steps in the software development and change process. The coverage of the FSED process also introduces the reader to the type of diagrams used to document the AISF's functional requirements.

The vehicle for documenting the functional requirements is the Data Flow Diagram (DFD). The DFD is a graphic tool used to depict the logical flow of data through a system. The DFD also known as a Bubble Chart or Data Flow Graph is particularly useful for identifying the functions of a system and the resultant data transformations. (Ref 34:55) The diagrams employed here each represent a certain level of detail within the AISF operation. Functional requirements are stated at a high level, then iterated through the DFDs to detailed functional requirements. The complete hierarchy is documented in Appendix A. Accompanying the DFDs are charts showing the hardware and software tools required to fulfill the functional requirements illustrated in the diagrams. Figure 10 shows an example of the format in which the functional requirements are presented. Complete functional requirements for all four phases of the OFP change process are presented in this manner.



Tools Required	
Hardware	Software
Hardware Required to Implement Functional Requirements	Software Required to Implement Functional Requirements

Figure 10. DFD Format

Following the functional requirements is a generic resources list for implementing an AISF. This list covers, at a relatively high level of abstraction, the resources required. The resources are presented in a hierarchical format and include hardware, software, physical facility, personnel, training, and documentation resources required to fulfill the functional requirements presented earlier.

Full Scale Engineering Development of Software

During the deployment or Operational and Support (O & S) phase of a weapon system life cycle, there will be recurring needs for making changes in operational programs. These changes may be required because of a change in the threat, mission of the weapon system, or changes in other aspects of the weapon system such as ECS hardware. Changes may result from correction of program deficiencies or problems, improvement in program operation, or enhancements to improve capability. The basic change cycle or sequence consists of the same steps required during the Full Scale Engineering Development (FSED) of the software.

The FSED process is performed in a sequence of steps which converts concepts into operational software and consists of the following generic group of software-specific activities: Requirements (analysis/definition), Preliminary Design, Detail Design, Code and Unit Test, Integration and Test, and System Acceptance. (Ref 5:9) The FSED of an operational flight program (OFP) can be depicted in DFD format as shown in Figure 11.

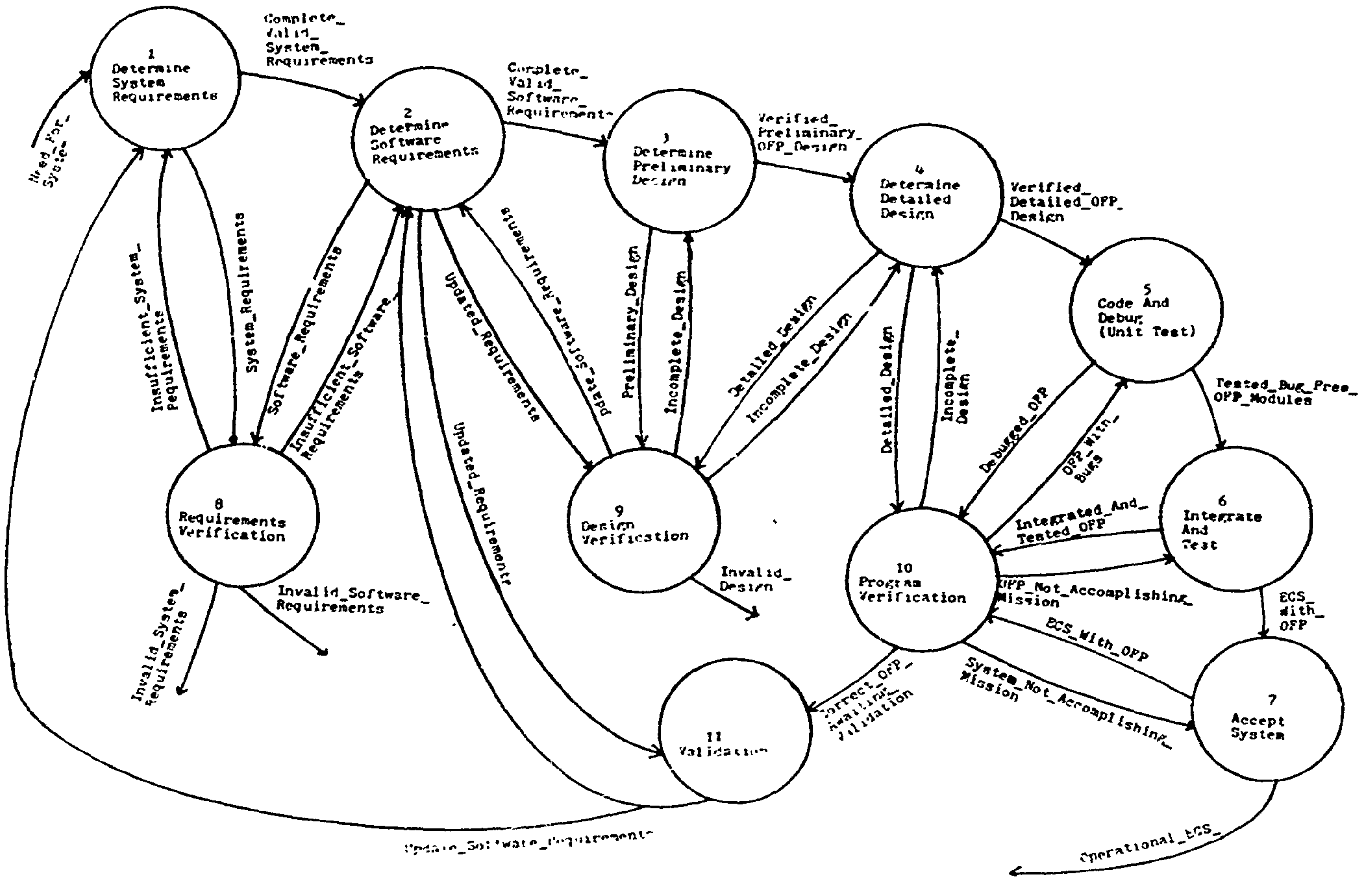
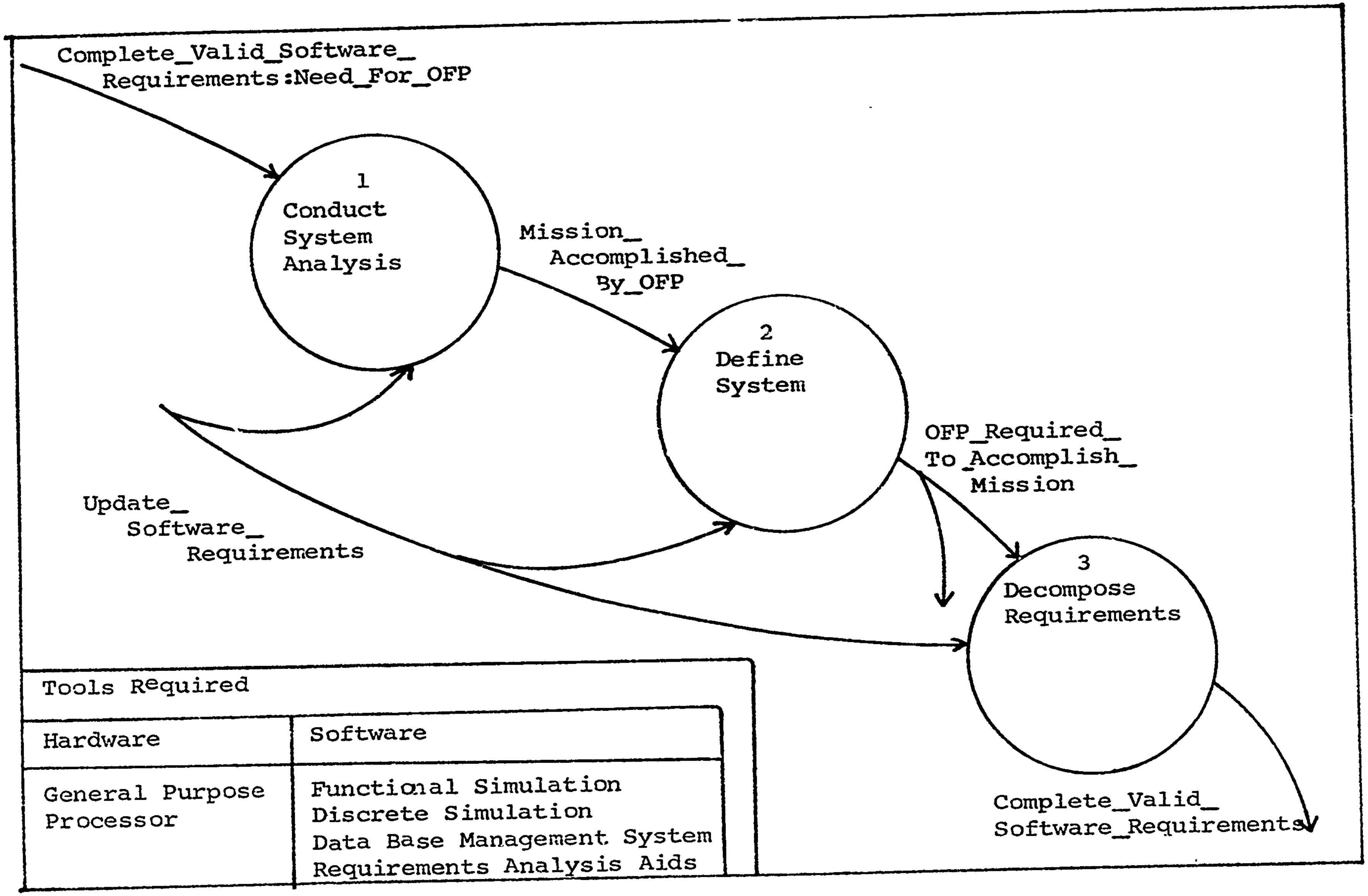

Need_For_System_
1 Determine System Requirements
Complete_Valid_System_Requirements
2 Determine Software Requirements
Complete_Valid_Software_Requirements
3 Determine Preliminary Design
Verified_Preliminary_OFP_Design
4 Determine Detailed Design
Verified_Detailed_OFP_Design
5 Code And Debug (Unit Test)
Tested_Bug_Free_OFP_Modules
6 Integrate And Test
ECS_With_OFP
7 Accept System
Operational_ECS_
Insufficient_System_Requirements
System_Requirements
Software_Requirements
Insufficient_Software_Requirements
8 Requirements Verification
Invalid_System_Requirements
Invalid_Software_Requirements
Updated_Requirements
Update_Software_Requirements
Preliminary_Design
Incomplete_Design
Detailed_Design
Incomplete_Design
9 Design Verification
Invalid_Design
Updated_Requirements
Detailed_Design
Incomplete_Design
Debugged_OFP
OFP_With_Bugs
Integrated_And_Tested_OFP
OFP_Not_Accomplishing_Mission
ECS_With_OFP
System_Not_Accomplishing_Mission
10 Program Verification
Correct_OFP_Awaiting_Validation
11 Validation
Update_Software_Requirements

Figure 11. FSED Process For An OFP

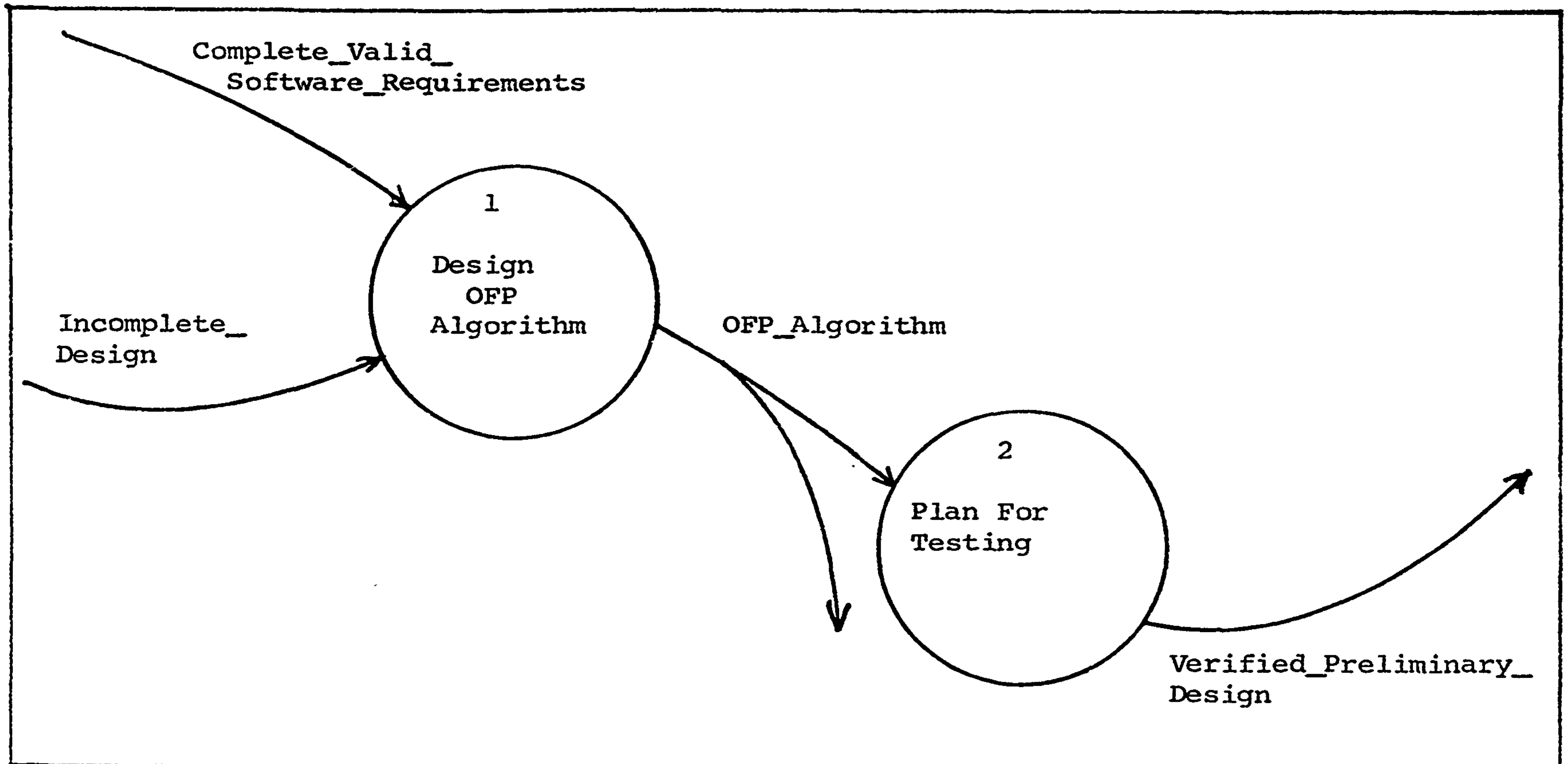


Tools Required	
Hardware	Software
General Purpose Processor	Functional Simulation Discrete Simulation Data Base Management System Requirements Analysis Aids

Figure 12. Determine Requirements

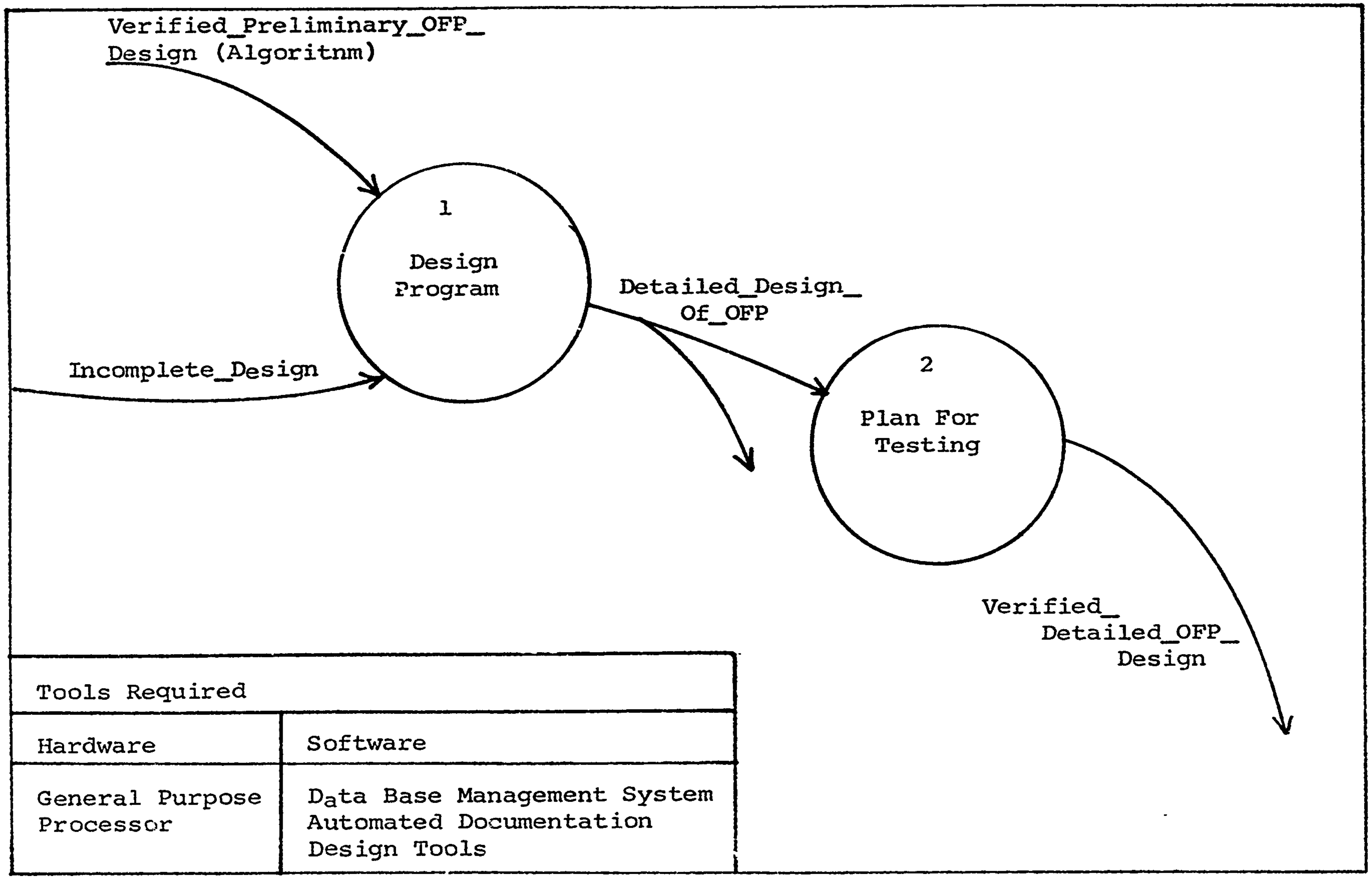
During the software development process, a set of software and hardware tools are assembled and used by the developer to effectively and efficiently accomplish each step. This collection of hardware and software is called a Software Development Facility (SDF) and it consists of the following three major categories of equipment/software: General Purpose Processor, Software Test Bed (STB), and Integration Test Bed (ITB) or Hot Bench. These three categories may be further subdivided into generic software and hardware support tools. The DFD of Figure 11 may be depicted in the next level of detail to illustrate the actions associated with each step of the FSED process of an OFP. Figures 12 through 17 illustrate this breakdown and accompanying tables list the generic categories of support hardware and software required to accomplish each step.

Figure 12 depicts the step of the FSED which produces the software requirements. During this step functional as well as discrete simulations of the system or subsystems are run on the General Purpose Processor. The General Purpose Processor is also used to host the Data Base Management System (DBMS) employed by the SDF and various requirements analysis aids. This general purpose large-scale computing capability is used throughout the FSED process to run off-line support software which is used to compile and assemble computer programs and to prepare the programs for loading into the flight computers. The General Purpose Processor may also host various analysis and design tools, data reduction and analysis programs, and the DBMS including configuration



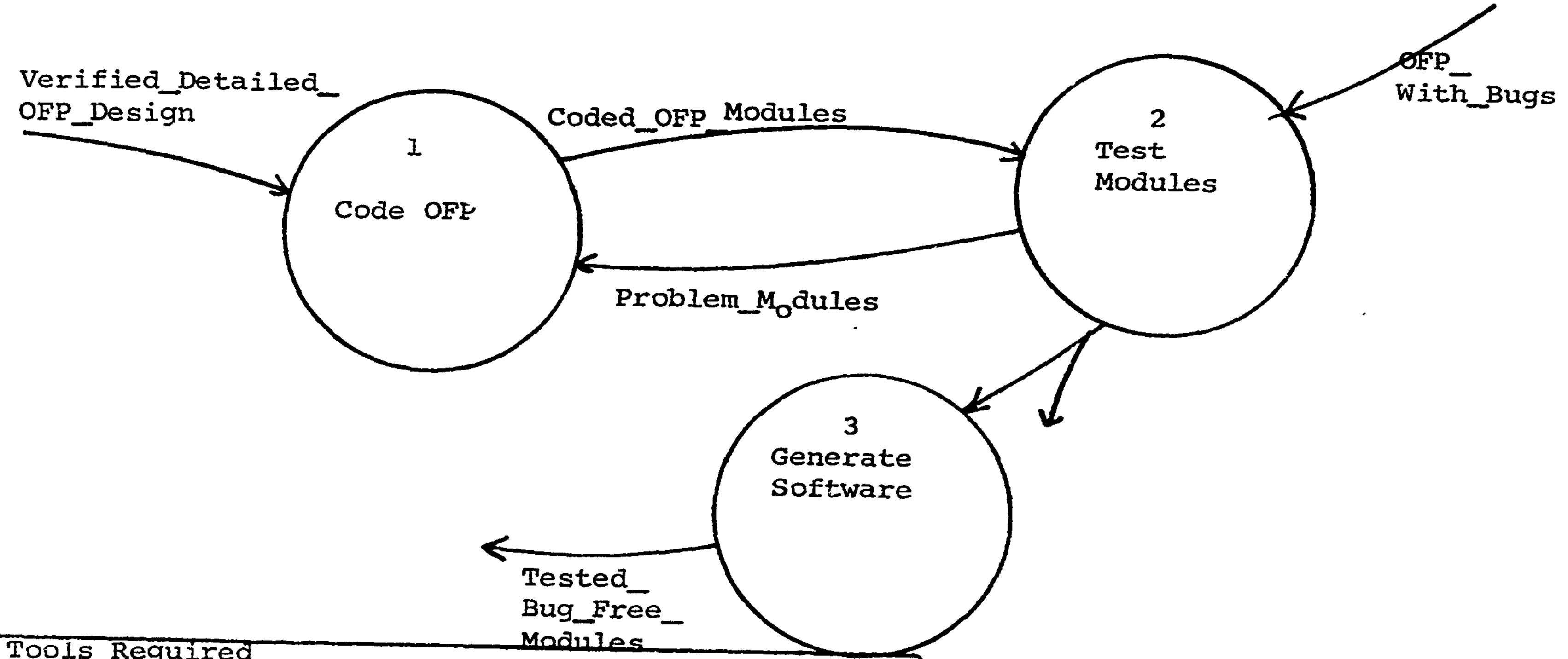
Tools Required	
Hardware	Software
General Purpose Processor	Scientific Simulation Data Base Management System Design Tools

Figure 13. Preliminary Design



Tools Required	
Hardware	Software
General Purpose Processor	Data Base Management System Automated Documentation Design Tools

Figure 14. Detailed Design



Tools Required	
Hardware	Software
Software Test Bed General Purpose Processor	Assemblers Compilers Loaders Scientific Simulations Emulators/Interpretive Computers Data Reduction and Analysis Simulation Software

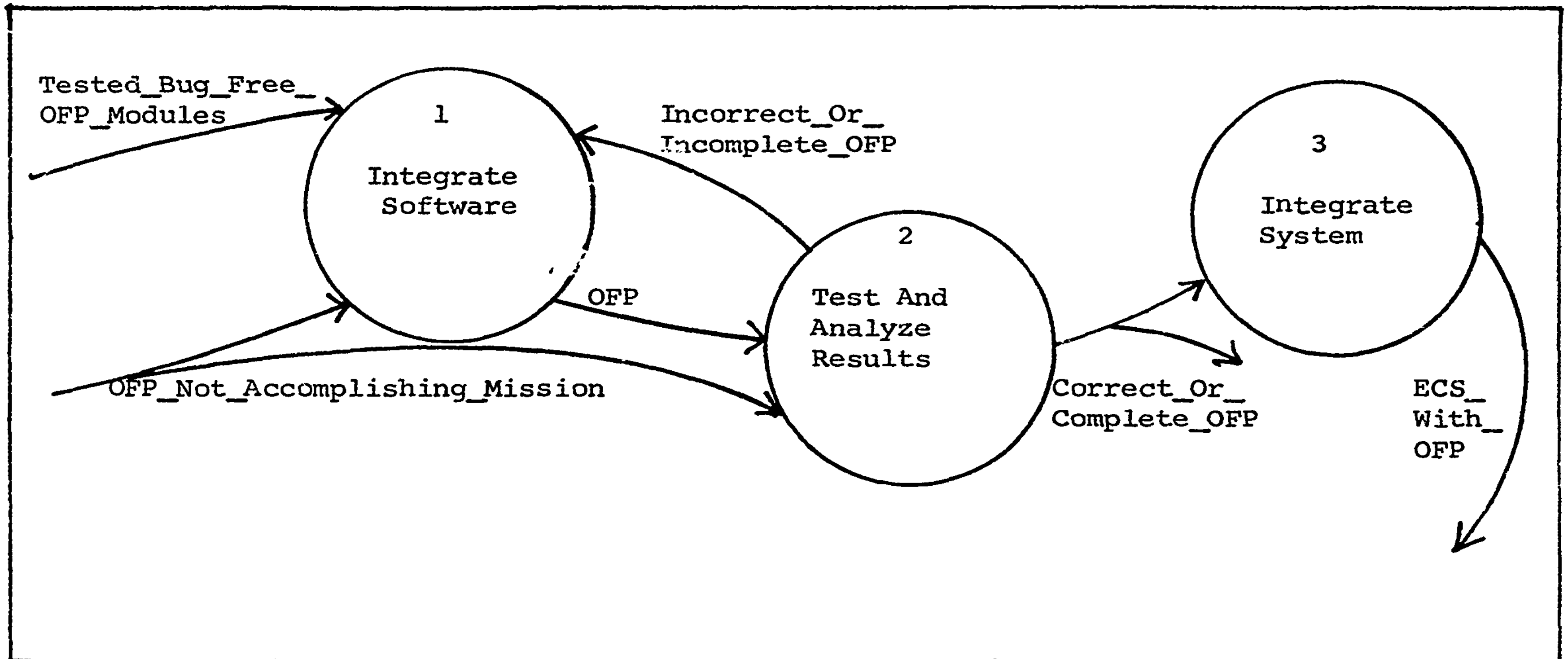
Figure 15. Code And Unit Test

control support.

In the Preliminary Design step of the FSED process (Figure 13), scientific simulations of the system are run on the General Purpose Processor in order to determine design parameters. The General Purpose Processor also hosts design tools used in designing the algorithm for the OFP as well as the DBMS used for configuration management. The product of the Preliminary Design Phase is an algorithm and a test plan for the OFP requirements generated in the Requirements Determination Phase.

In the Detail Design Phase (Figure 14), the algorithm produced in the previous phase is transformed into an actual program design. The General Purpose Processor hosts design and automated documentation tools for this purpose. The DBMS is also used for configuration management. The products of this phase are a detailed design for the OFP as well as a test plan for testing the design.

The Code and Unit Test Phase of the FSED process (Figure 15), involves actually writing and testing the code comprising the OFP modules. In this step, the detailed design produced in phase four is transformed into tested, bug-free modules ready for integration into a complete OFP. The Code and Unit Test process requires the General Purpose Processor for hosting the necessary assemblers, compilers, and loaders required to produce the modules. This phase also requires the use of the Software Test Bed (STB) which is used to run the operational program modules on the actual operational (Avionic) computer, while another computer is used to simulate the inputs to the interfaciing electronic



Tools Required	
Hardware	Software
Software Test Bed Integration Test Bed General Purpose Processor	Assemblers Compilers Loaders Data Reduction and Analysis Data Base Management System Simulation Software

Figure 16. Integrate And Test OFP

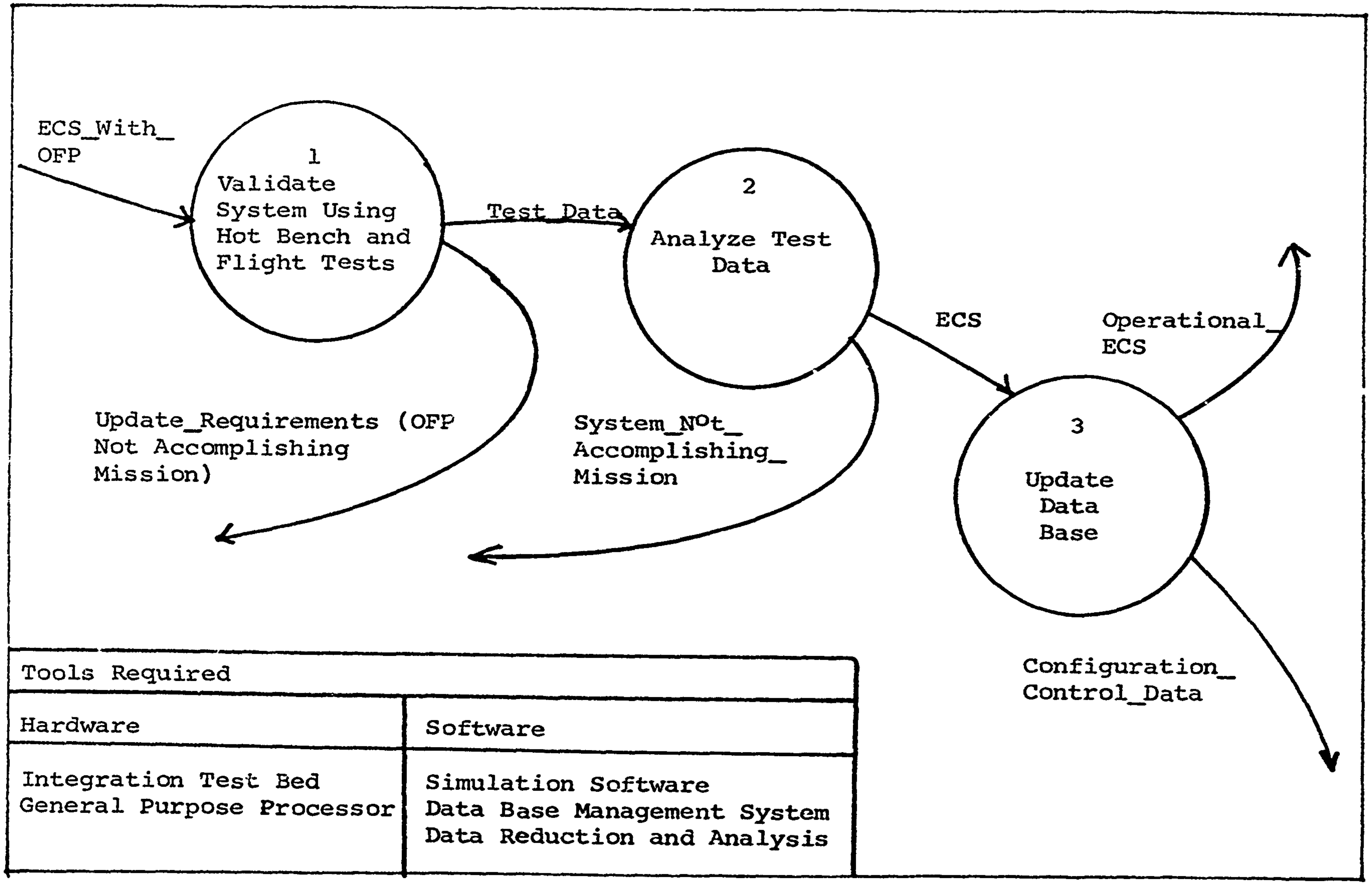
equipment. Program module pe:formance evaluation is obtained by monitoring the operational computer memory and electronic interfaces. The analyst should be able to monitor selected registers and addresses of the operational computer and have the capability to stop the program, change parameters, restart the program, and single step through the program. The capability should also exist for the analyst to perform traces, traps, and dumps on selected areas of the OFP.

In the Code and Unit Test Phase of the FSED process, scientific simulations are used to provide sensor inputs to the operational program modules and simulations may be used to emulate the operational computer when it is not available. The emulator provides the capability for developing the software modules before the operational computer has been completed.

The end product of a successful Code and Unit Test Phase is a collection of tested and bug-free modules to be integrated into an OFP.

During the Integration and Test Phase (Figure 16), the modules are integrated into a complete OFP and tested. The General Purpose Processor and STB are used along with compilers, assemblers, loaders, and simulation software to accomplish the integration. Once a completely integrated OFP is obtained it is tested using Data Reduction and Analysis tools running on the General Purpose Processor.

After a complete and correct OFP is obtained it is integrated into the system (ECS) using an Integration Test Bed or Hot Bench. The ITB is similar to the STB except that system equipment is used in addition to the operational computer.



Tools Required	
Hardware	Software
Integration Test Bed General Purpose Processor	Simulation Software Data Base Management System Data Reduction and Analysis

Figure 17. System Acceptance

For example, instead of simulating the electronic interface equipment on a computer, the actual MIL-STD 1553 Multiplex Bus would be used in the ITB. The equipment is again artificially stimulated with scientific simulations and mathematical models so that it produces realistic inputs to the operational computer. The ITB is a tool used specifically for equipment/OFP integration and the product produced from this integration is an Embedded Computer System with an OFP ready for the final phase of the FSED process, System Acceptance (Figure 17).

In the final phase of the FSED process, System Acceptance, the system is validated with flight tests and on the ITB. Data Reduction and Analysis tools hosted on the General Purpose Processor are used to analyze results of the tests to determine whether the system is functioning properly. If the system successfully passes the validation portion, the data base for configuration management is updated and an operational ECS is completed. If the system fails the validation process, the system requirements and/or software requirements must be reexamined and the FSED process reaccomplished.

Software Support Activities and Requirements

Software updates and ECS prototypes and modifications are performed with detailed planning and are carried out in an orderly, systematic, and well-controlled process which parallels the FSED process discussed in the previous section of this chapter. Software (OFP) updates are viewed as microcosms of the acquisition development process and a

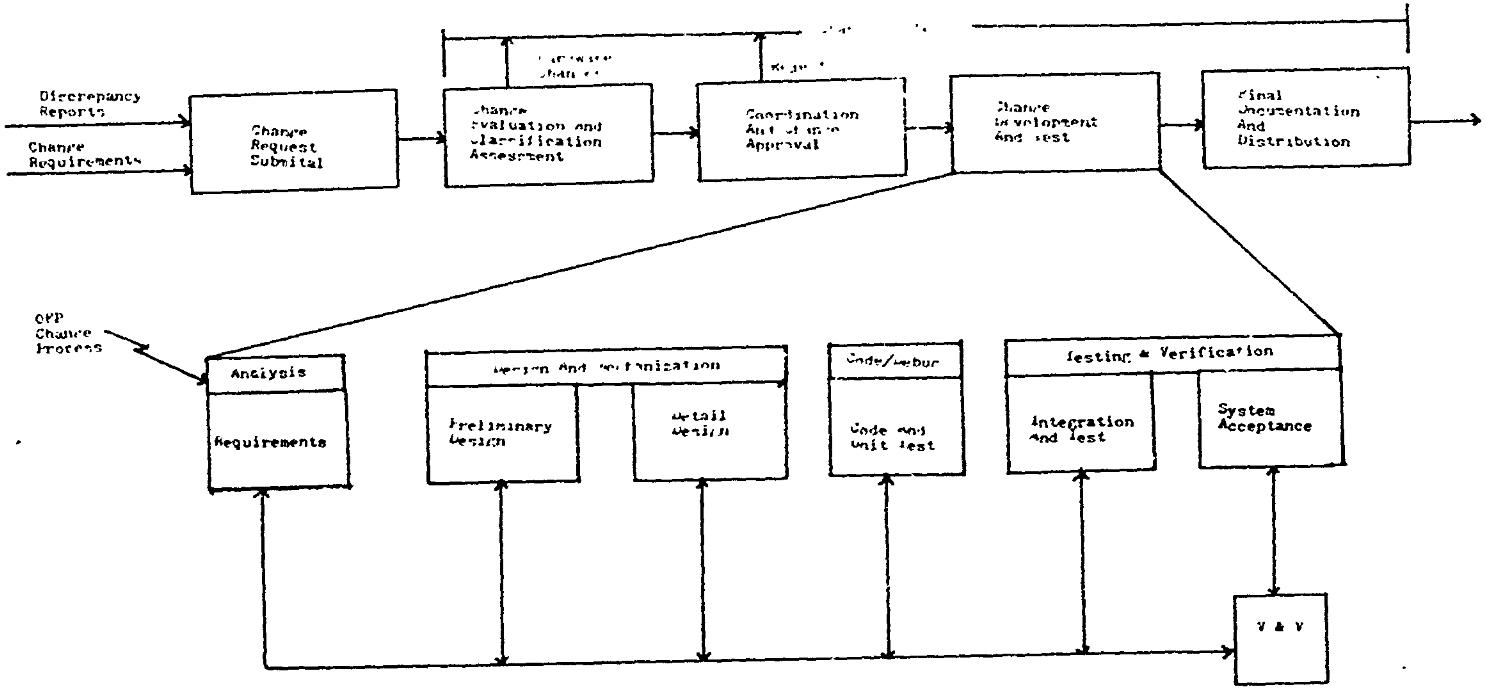


Figure 18. Software Change Process During Operations And Support

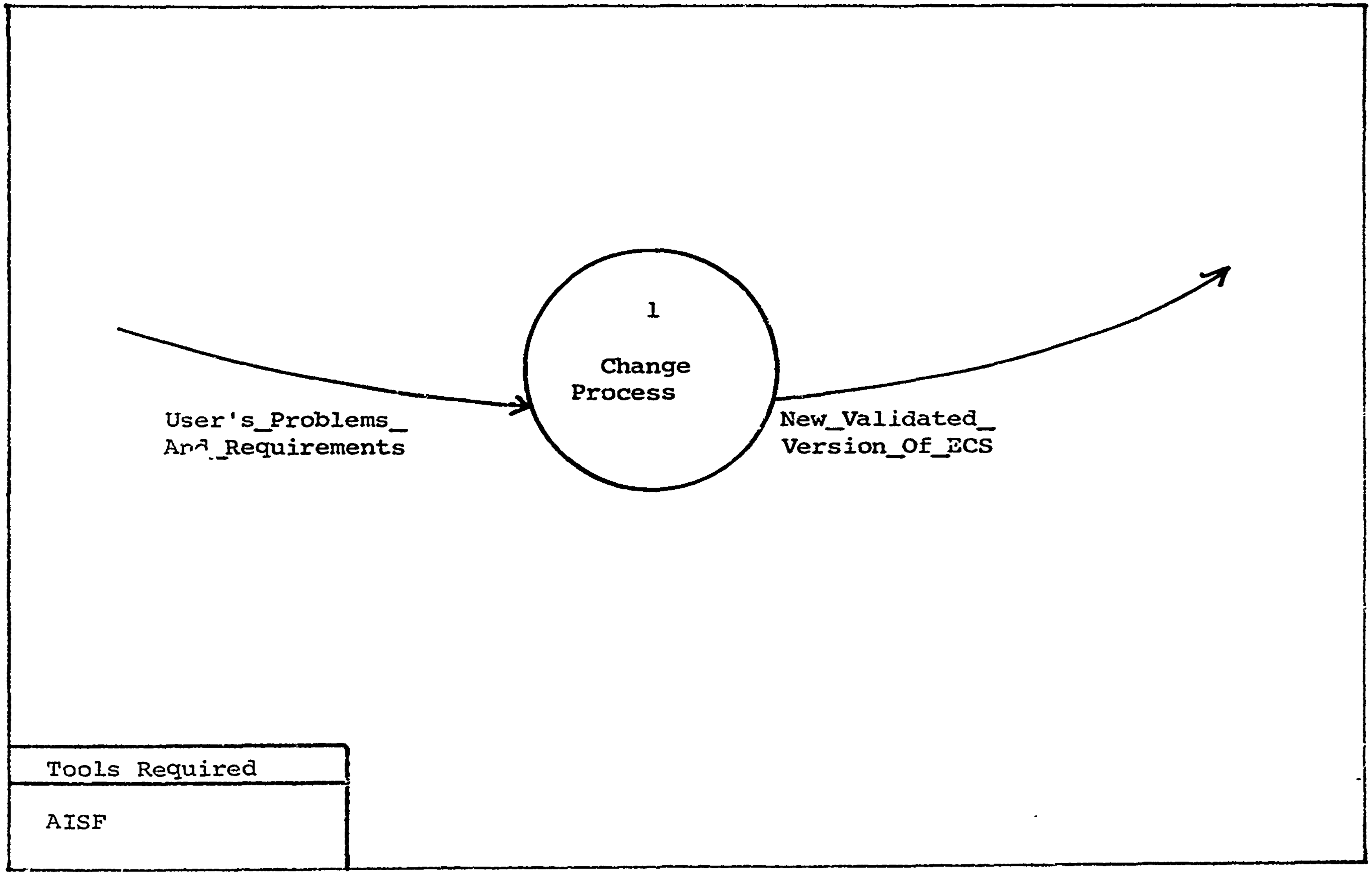


Figure 19. OFP Change Process

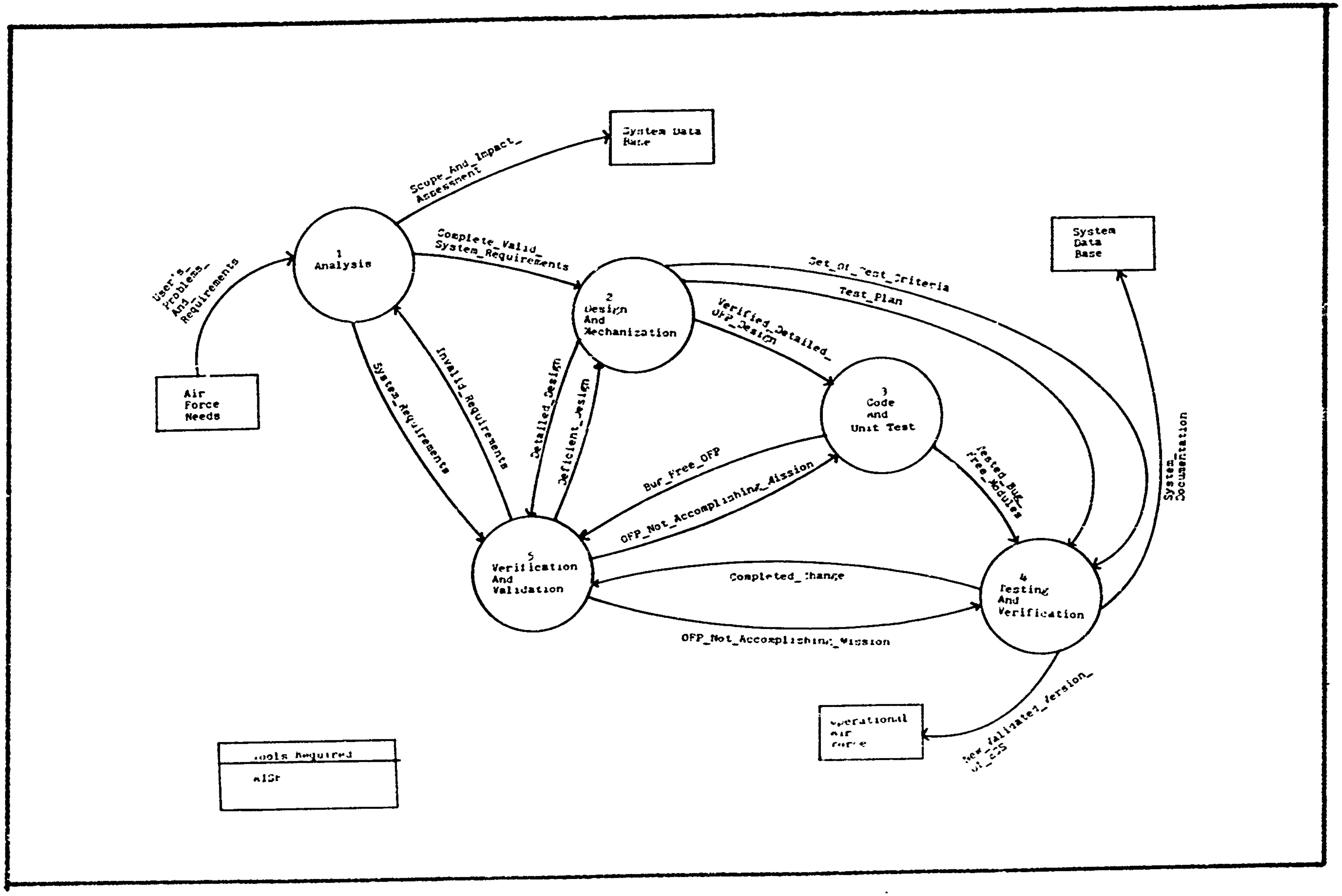

Air Force Needs
User's_Problems_And_Requirements
1 Analysis
Scope_And_Impact_Assessment
System Data Base
Complete_Valid_System_Requirements
2 Design And Mechanization
Set_of_Test_Criteria
Test_Plan
System Data Base
Verified_Detailed_OFP_Design
3 Code And Unit Test
System_Requirements
Invalid_Requirements
Detailed_Design
Deficient_Design
Bug_Free_OFP
OFP_Not_Accomplishing_Mission
Tested_Bug_Free_Modules
System_Documentation
5 Verification And Validation
Completed_Change
OFP_Not_Accomplishing_Mission
4 Testing And Verification
New_Validated_Version_of_OFP
Operational Air Force
Tools Required
AISF

Figure 20. Steps In OFP Change Process

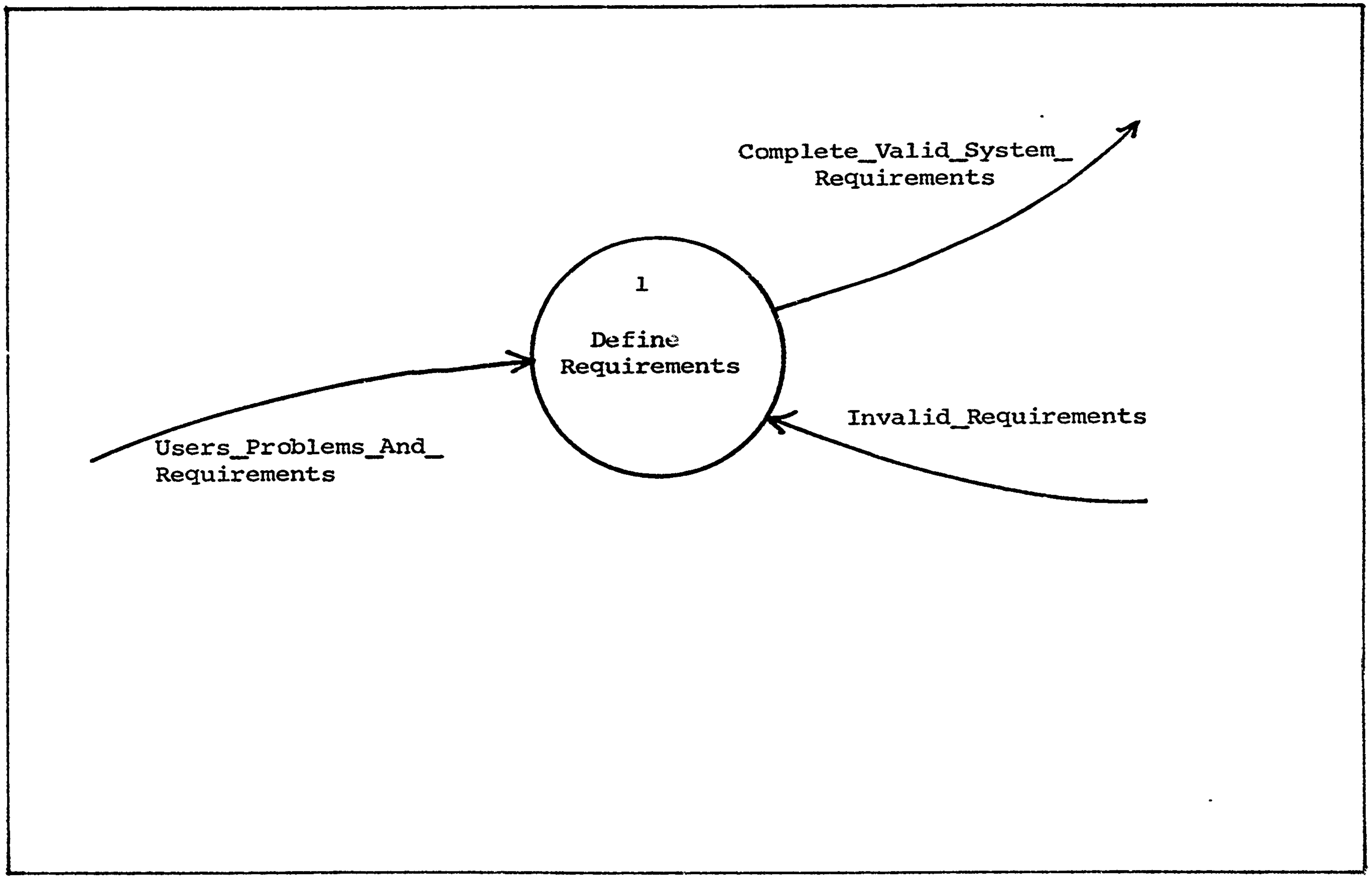
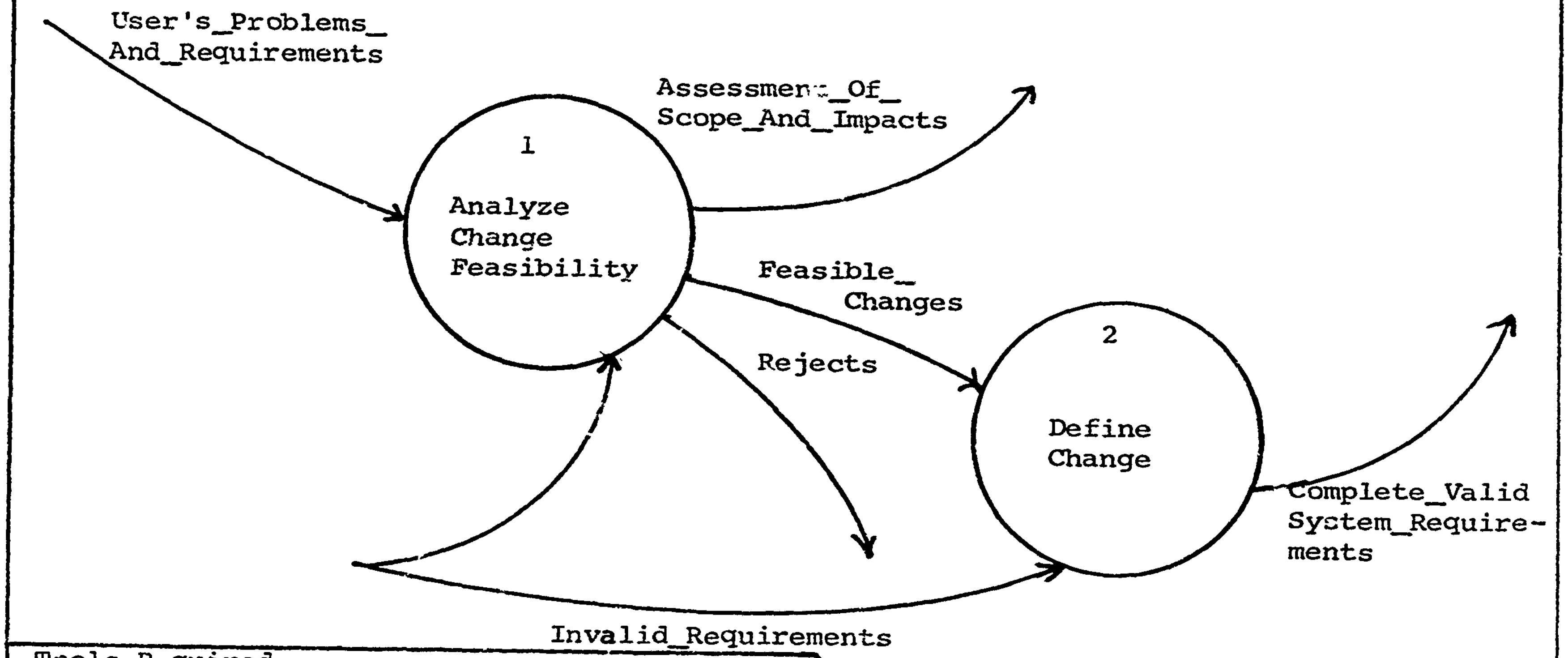


Figure 21. Analysis Phase-Requirements Definition

routine change cycle will be approximately 12-18 months in duration. Steps within this process utilize actions which are the same as those comprising the FSED process: Requirements (Analysis/Definition), Preliminary Design, Detail Design, Code and Unit Test, Integration and Test, and System Acceptance. Figure 18 shows the steps involved inthe software change process and where the process fits into the overall ECS change cycle. The software change process begins with the analysis of user's problems and requirements, and culminates with a new, validated version of the ECS. The tool used to accomplish the changes is the AISF. Figure 19 shows the highest level DFD illustrating the OFP change process. Figure 20 shows the next level of detail, the fact that the change process may be broken down into four steps: Analysis, Design and Mechanization, Code and Debug, and Testing and Verification. These are the high-level functions of the AISF.

Analysis Phase of OFP Change Process. The change process begins with the Analysis step which consists of the requirements definition. Analysis of problems is performed using both the General Purpose Processor and the STB. During this phase, a determination is made regarding validity, feasibility, risks, impacts, and scope of proposed changes. Table 2 (Ref 23:130) lists considerations during the Analysis Phase. In the DFD of Figure 21 the fact that the OFP change Analysis Phase is analagous to the FSED Requirements Definition Phase is illustrated. Figure 22 breaks down the Analysis Phase into yet another level of detail. Those considerations given in Table 2 contribute to the feasibility study.



Tools Required	
Hardware	Software
General Purpose Processor	Data Base Management System Requirements Analysis Aids Functional Simulation

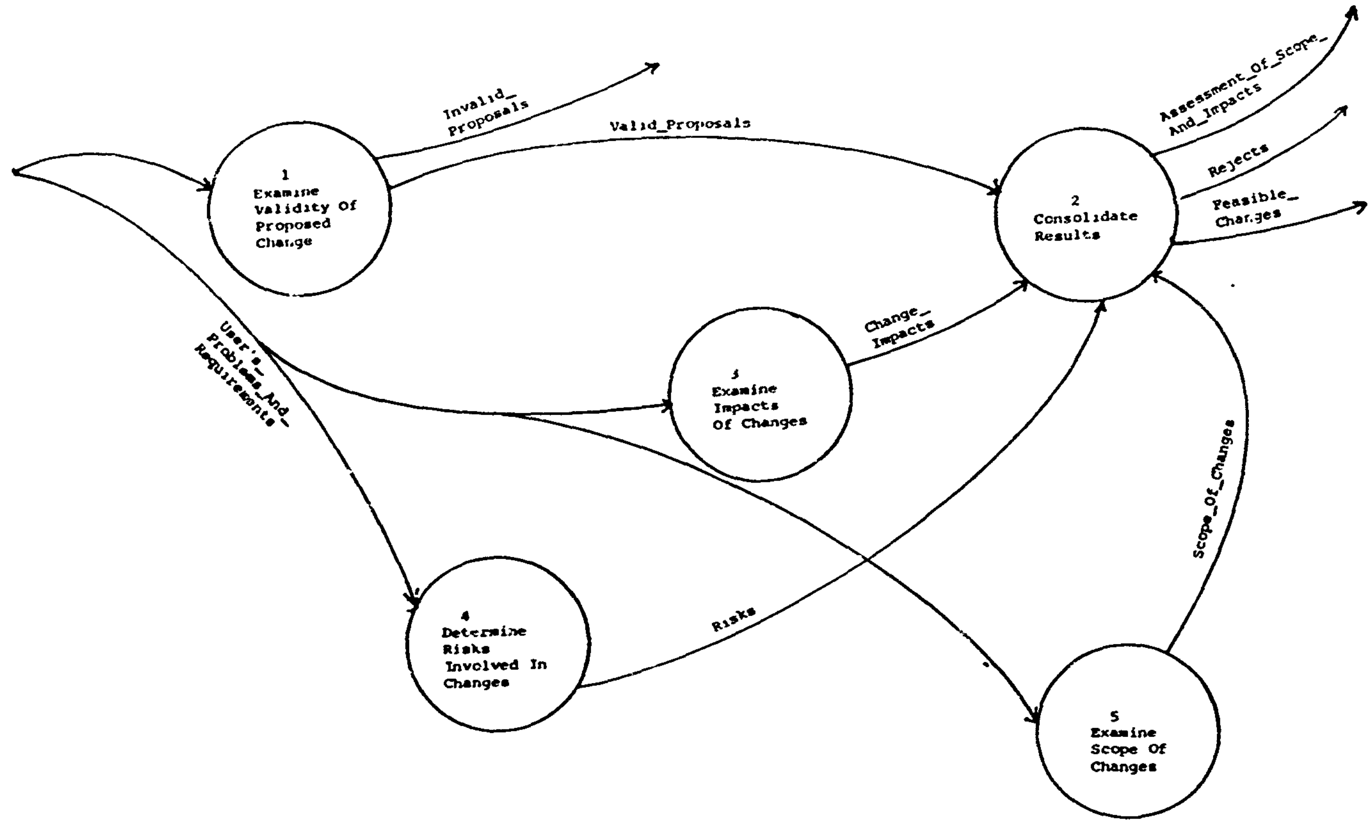
Figure 22. Analysis Phase-Feasibility Study

Table 2

Considerations During Analysis Phase

- Operational performance and interface and functional requirements
- Computer time, speed, accuracy, and storage requirements
- Software structure
- Operator workload impacts
- Interfaces and weapon system impacts
- Test provisions and data
- Useability, maintainability, and reliability
- Documentation requirements
- Integration and acceptance testing
- Time, cost, and resource requirements
- Alternatives
- Tradeoff considerations

The feasibility study can be further divided into the functions illustrated in Figure 23. The General Purpose Processor is the main hardware used to accomplish the functions of the feasibility analysis and change definition activities. Software tools hosted by the General Purpose Processor include a DBMS,requirement analysis aids, and functional simulations. The DBMS is used for managing files associated with the feasibility study and change definition. A transaction log should be maintained documenting all file activities and providing management information concerning the status of the Analysis Phase functions, and verification and validation activities. The DBMS should include report generation



Tools Required	
Hardware	Software
General Purpose Processor	Data Base Management System Requirements Analysis Aids Functional Simulation Discrete Event Simulations

Figure 23. Analyze Change Feasibility

capabilities so that data concerning the status of the Analysis Phase can be presented in a management-readable format. Data bases hosted on the General Purpose Processor and managed by the DBMS of importance to this phase include Engineering Change Proposal (ECP) Status, OFP Status, and AISF Configuration.

The analysis support software hosted on the General Purpose Processor are intended to aid the engineer/programmers in performing system engineering, feasibility analysis, verification and validation, and problem analysis. The ECS requirements are maintained for performing traceability analysis, analyzing their completeness, and identifying their interactions during the Analysis Phase. Requirements tracing should be automated by applying appropriate formal documentation rules. A partial list of requirements analysis aids is provided by Table 3.

Table 3

Requirements Analysis Aids

- Data Flow Diagrams
- Q-GERT Modeling and Simulation Language
- Decision Tables/Trees
- Structure Charts
- Structured English
- Flowcharters
- Automatic Work Control Systems
- Automated Drafting Tools

The Functional Simulation is another software tool utilized by the AISF during the Analysis Phase. Functional Simulations are grossly simplified analytic models programmed on the General Purpose Processor. They model a portion of the overall system or an aspect of the system's behavior, and are used for the requirements definitionportion of the Analysis Phase. Another tool, the Discrete Event Simulation, may be used to model gross system configuration as a function of specific events on the mission time line. Discrete Event Simulations are used to compare target computer capabilities to maximum throughput requirements and to evaluate executive design tradeoffs. The network modeling vehicle Q-GERT may be useful for Discrete Event Simulations.

Often, in the AISF environment, some of the functions which are normally run on the General Purpose Processor can be combined in the processor of the Software Test Bed (STB) or the Simulation Host Processor (SHP). The STB is primarily used for the Code and Debug and Testing and Verification Phases of the OFP change process. Discussion of the STB is thus deferred to those sections of this chapter.

The output of the Analysis Phase of the OFP change process is a definition of change requirements and an assessment of scope and impacts.

Design and Mechanization Phase. During the Design and Mechanization Phase, the requirements generated during the Analysis Phase are translated into a specification that states interface, performance, and functional requirements and further translated into a detailed design which defines program structure, logic, timing, inputs, outputs, algorithms, and data bases. (Ref 23:131)

The design and Mechanization Phase of the OFP change

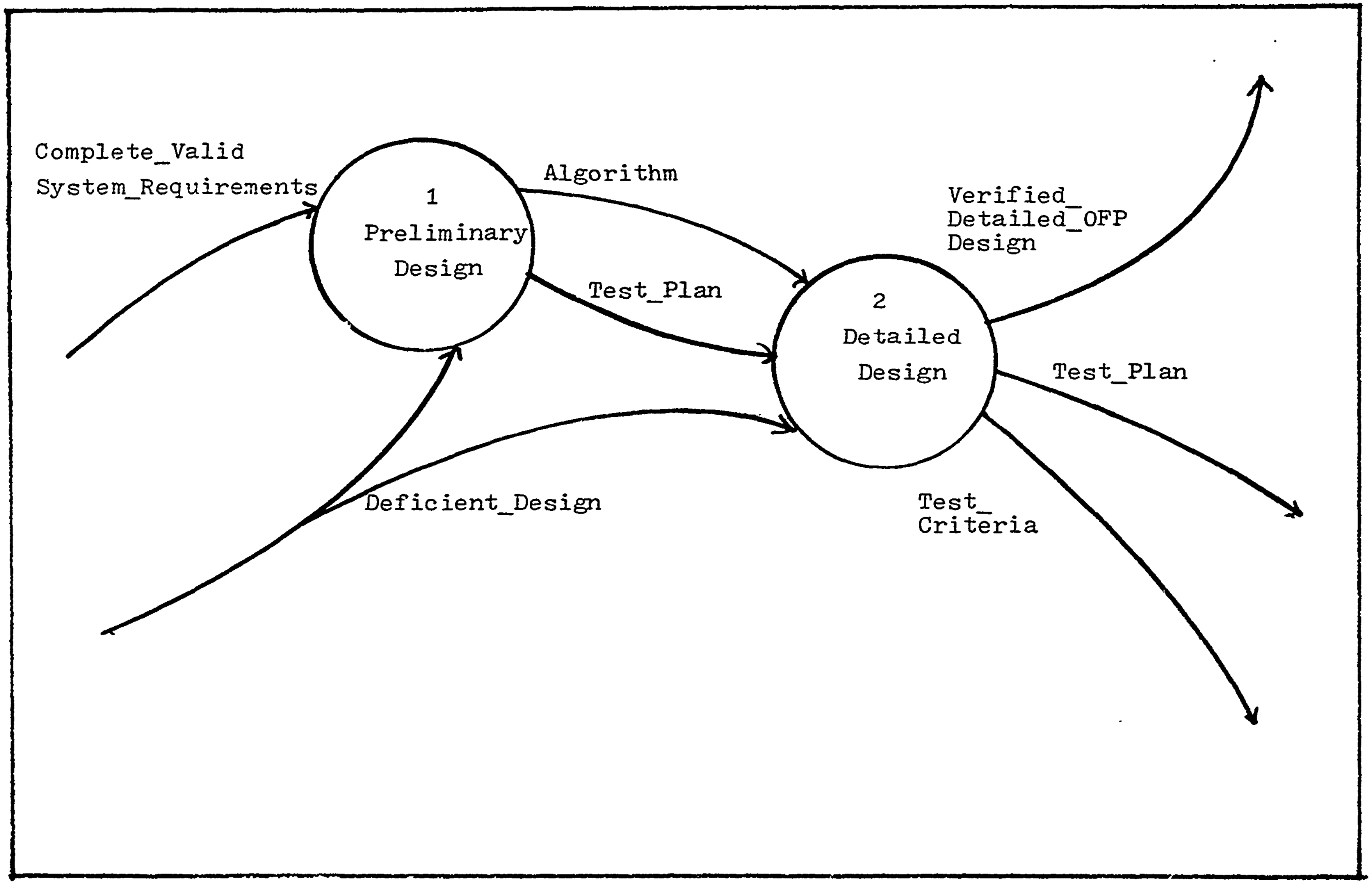
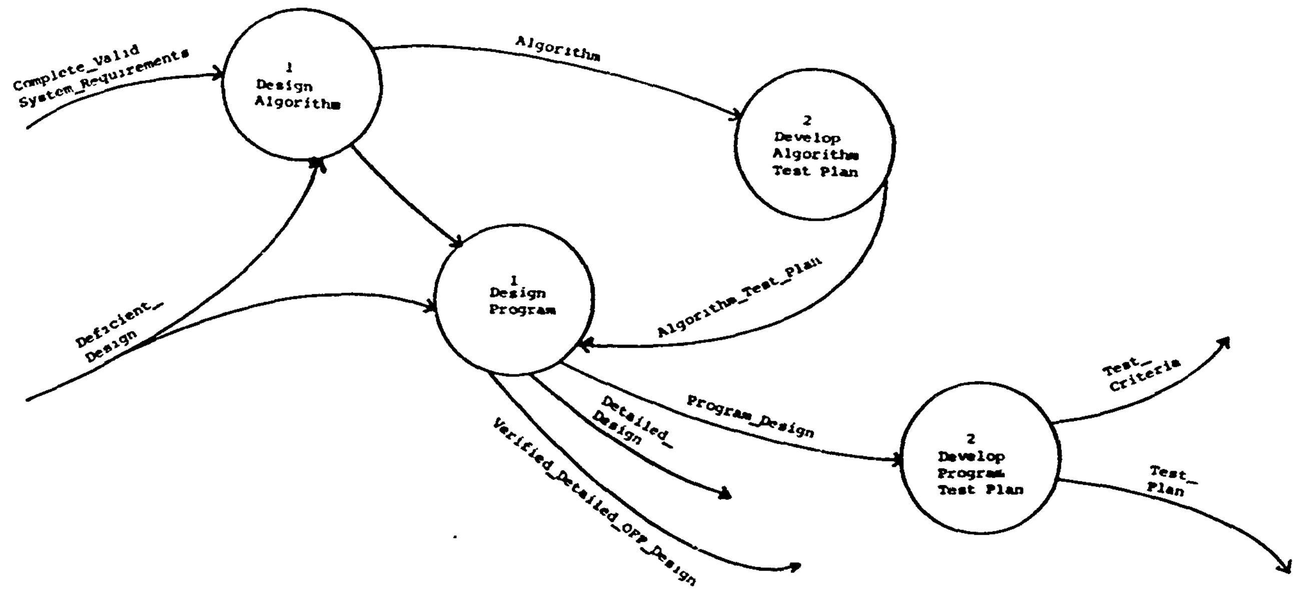


Figure 24. Design And Mechanization Phase-Second Level Of Detail



Tools Required	
Hardware	Software
General Purpose Processor Software Test Bed	Scientific Simulations Data Base Management System Design Tools

Figure 25. Design And Mechanization Phase-Third Level Of Detail

cycle encompasses steps analogous to the Preliminary and Detail Design steps of the FSED process. The outputs of the Preliminary Design step are an algorithm for the proposed OFP change and a test plan for testing the algorithm. These outputs serve as inputs to the Detail Design step which produces the verified and detailed OFP design, a test plan for the design, and a set of test criteria to be used in the Test and Verification Phase. Figure 24 shows the highest level of detail for this phase.

Figure 25 shows the next level of detail in the Design and Mechanization Phase. The functions performed during the Preliminary Design and Detail Design steps are shown as well as hardware and software tools required to accomplish the functions.

The first action accomplished during the Preliminary Design portion of the Design and Mechanization Phase is transforming the system requirements from the Analysis Phase into an algorithm. Design tools, data base management, and scientific simulations are run on the General Purpose Processor for this purpose.

The DBMS is again used in the Design and Mechanization Phase for files management and report generation. A transaction log should be maintained and management information provided concerning the status of the software development and the verification and validation activities. The DBMS should be used to track OFP status, and the AISF configuration during this phase of development.

Various design tools hosted on the General Purpose

Processor should be employed during the Design and Mechanization Phase. There is a wide spectrum of available design tools some of which are listed in Table 4.

Table 4

Design Tools

- Data Flow Diagrams
- Structured English
- Decision Tables/Trees
- Flowcharting and Flowcharters
- HIPO and Function Charts
- Structured Analysis and Design Technique (SADT)
- Module Coupling and Cohesion Analysis
- Scientific Simulations
- Q-GERT Simulations
- Logic and Equation Generators
- Automatic Work Control Programs
- Automated Drafting Tools

A scientific simulation is utilized during the Design and Mechanization Phase to represent the equations and logic of the target avionics computer software, but not the features of scaling, timing, and overflow conditions, or the details of the software/hardware interfaces. The system hardware and environment are represented by application software in the form of mathematical models. The scientific simulation is used during the Design activity to verify design concepts and to investigate the effects of software and/or hardware changes on system performance. Another use for the scientific simulation is for the prediction of system performance and accuracy of specified system hardware tolerances. The

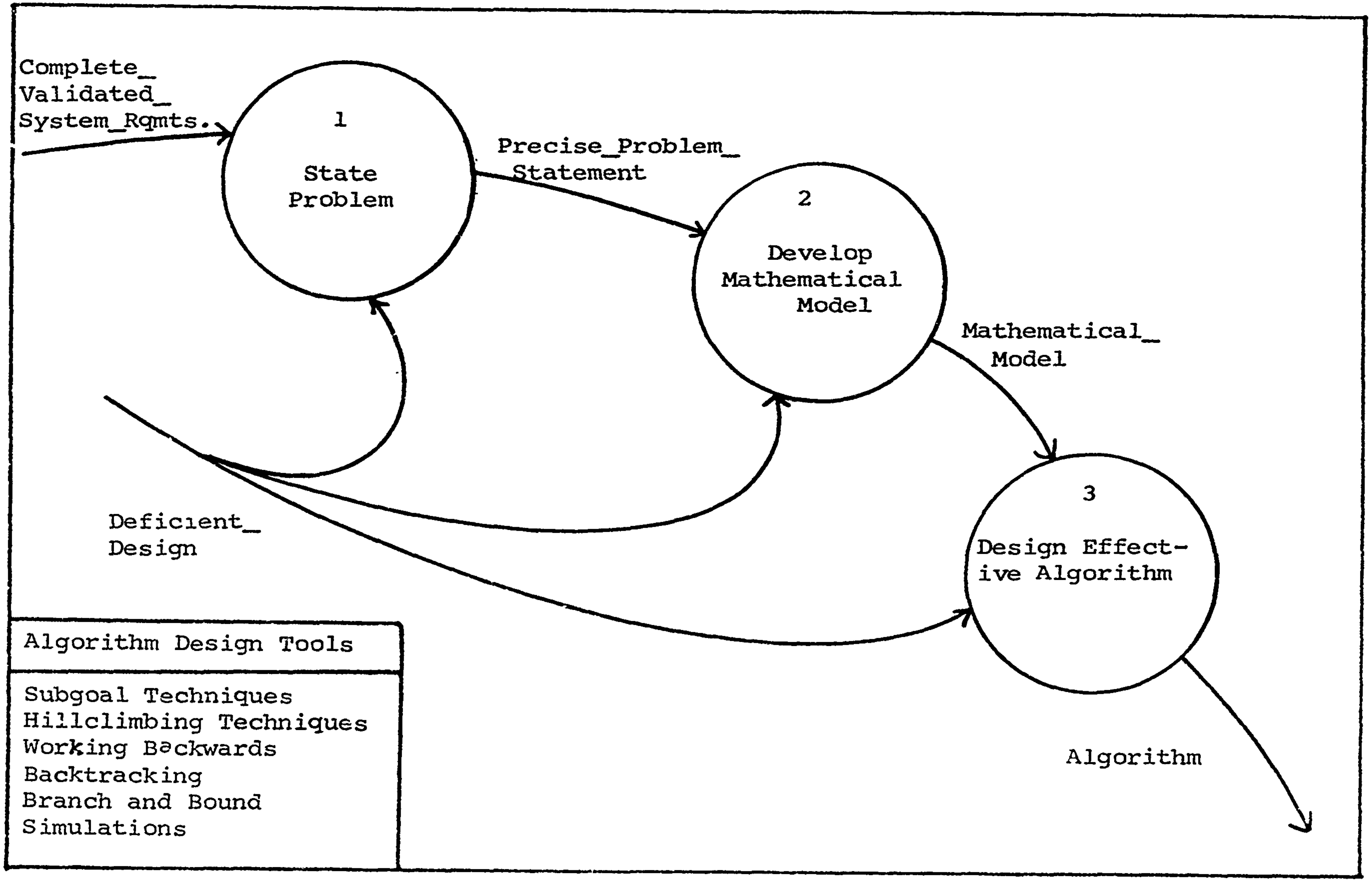


Figure 26. Design Algorithm

Q-GERT simulation and analysis tool is also a particularly useful tool for verifying deisgn and predicting system performance. An implementation using this technique is presented in Chapter VI of this report.

During the Design and Mechanization Phase, the development of an algorithm test plan is also conducted. The algorithm and test plan serve as inputs to the process to actually design the new OFP.

The algorithm and algorithm test plans are inputs to the Program Design function. During this portion of the Design and Mechanization Process a detailed program design is produced. Concurrently, a program test plan and a set of program test criteria should be developed. Many of the design tools listed in Talbe 4 are used in the Program Design Process.

A further breakdown of the functions performed during the Design Algorithm process is shown in Figure 26. System requirements provide the framework upon which a precise problem statement is developed. A precise problem statement should tell the algorithm designer exactly which information is available and what still needs to be found. Once a precise problem statement is formulated, it must be modeled mathematicallly. There are at least two basic questions to be addressed when formulating this model. First, the modeler must ask, which mathematical structures seem best suited for the particular problem, and, are there other problems that have been solved which resemble this problem? Another important consideration in the design of an algorithm is its effectiveness. Justification should be offered for each step of the algorithm involving a lemma about conditions that exist before and after the step is executed. (Ref 14:9) An attempt should be made to offer proof that the algorithm will terminate and in doing so, will have examined all the appropriate input data and produced all the appropriate output data.

Useful algorithm design tools and techniques are also listed on Figure 26. The subgoals approach to algorithm design involves reduction of a difficult problem into a sequence of simpler problems with more tractable solutions which can ultimately be combined to solve the original problem. Another technique of algorithm design, called hill climbing, makes an initial guess at a solution to the problem and then proceeds to move uphill from the initial solution toward better solutions as quickly as possible. Working backward involves starting at the goal or the solution to the problem and working backward toward the initial problem statement. A heuristic algorithm is one having the following properties: it will usually find good, though not optimal solutions and it is faster and easier to implement than any known exact algorithm. Heuristics may be used in the early stages of algorithm design and built up to obtain an optimal solution to the problem. The algorithm design technique called backtrack programming involves an organized, exhaustive, search which often avoids searching all possibilities. The backtrack programming technique is generally suitable for solving problems where a potentially large, but finite, number of solutions must be inspected. The design technique called branch and bound is similar

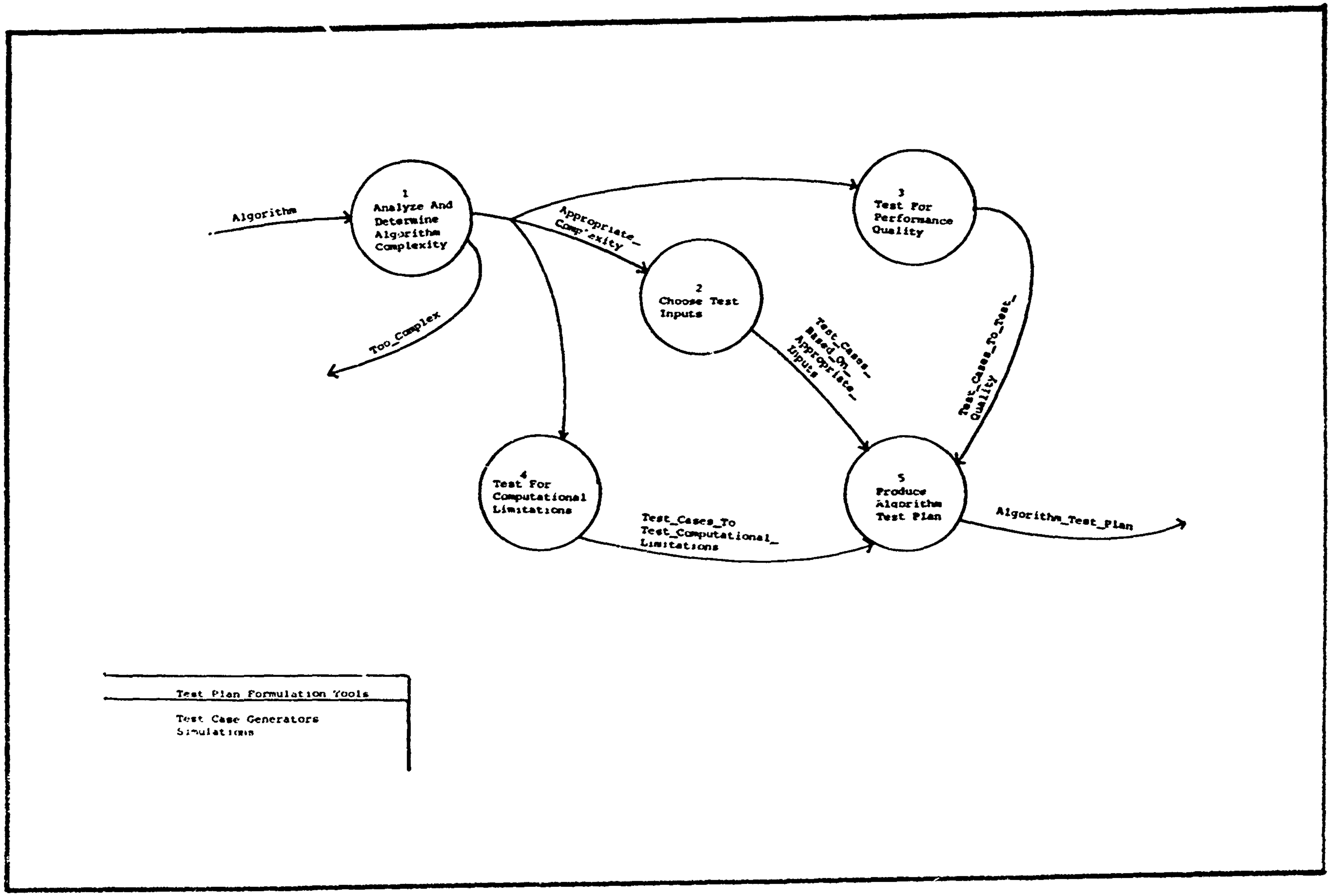


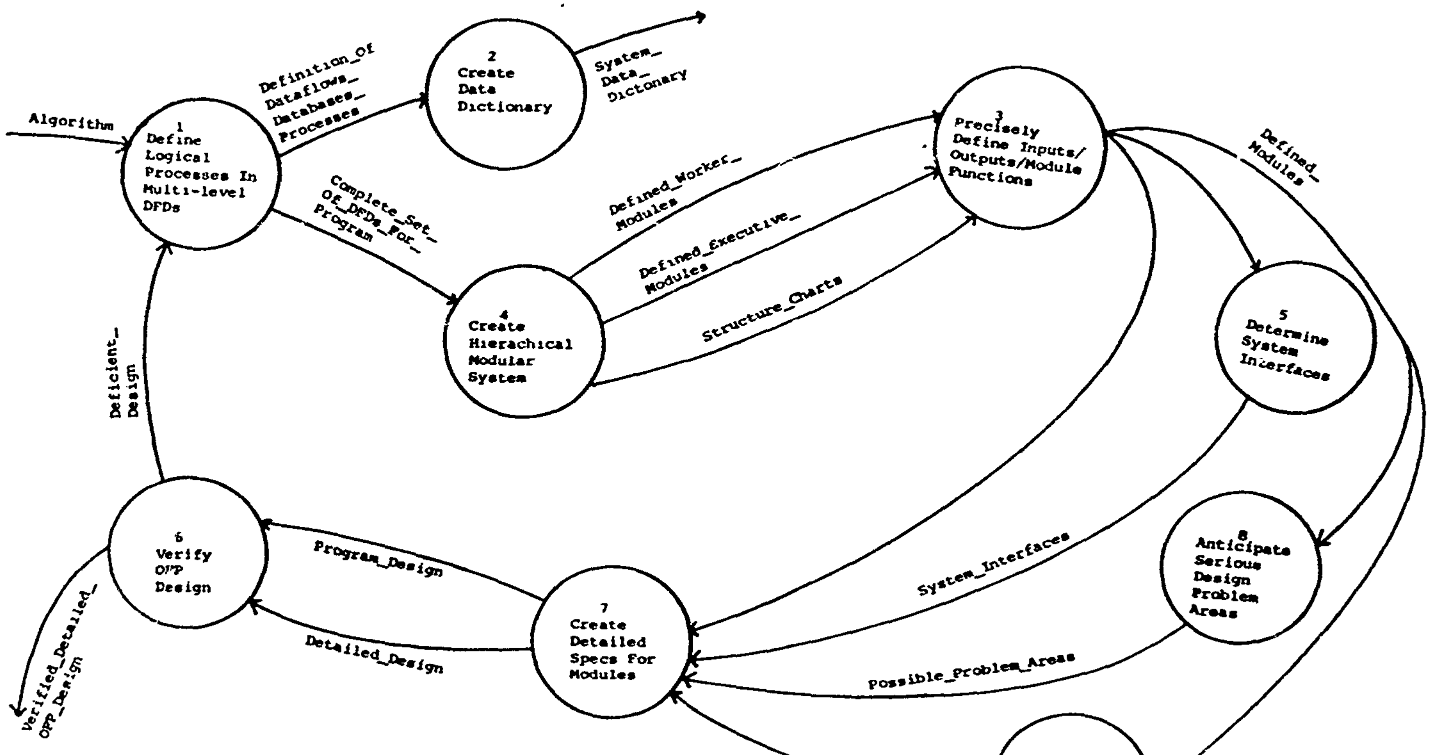
Figure 27. Develop Algorithm Test Plan

to backtracking in that it searches a tree model of the solution space and is applicable to a wide variety of discrete combinatorial problems. Branch and bound algorithms are oriented towards optimization. A search tree is constructed and the problem being solved specifies a real-world cost function for each of its vertices. The goal is to find a configuration which maximizes or minimizes the cost function. (Ref 14:96-134)

A technique called computer simulation is especially useful in the development of algorithms solving problems pertaining to large systems. Computer simulation is the process of conducting experiments on a computer model of a dynamic system with the immediate purpose of examining the behavior of the system under a given set of assumptions, parameters, and conditions. Simulations enable the designers to study all parts of a fully integrated system in as much detail as desired, whereas analytically, only individual parts of a system can be studied. Simulations allow for the control and measurement of all variables and enable the designer to obtain information about a real system when direct experimentation on the system is impractical. They also allow the designer to subject the system to different time scales. Time can be slowed to examine features on a microscopic level that are difficult to analyze in a real system, or it can be sped up to gain insight on a macroscopic level. Time can also be stopped, reversed, and replayed to study unusual behavior patterns in greater detail.

Testing may be described as an experimental verification that the algorithm is doing what it should. The algorithm test plan should be formulated with this fact in mind. The algorithm must be verified for a broad spectrum of inputs. Generally, the set of all possible inputs is prohibitively large and complete testing is impractical, therefore, a variety of inputs which test every aspect of the algorithm must be determined. A representative sampling of the cases to be encountered during system operation is a must. Test cases must also be formulated to ascertain the algorithm's performance quality. Simplifying assumptions made during the Requirements Analysis must be verified experimentally. Since many large, complicated algorithms are difficult to study mathematically, an experimental performance evaluation is especially important, since this is the only way to judge the quality of the algorithm. Producing analytical and experimental evidence of algorithm quality should be an integral part of the algorithm test strategy. The algorithm should also be tested for its computational limitations. It is desireable that the algorithm's performance change gracefully from good to bad as inputs change from good to bad. Test cases developed should include testing for the algorithm's computational limitations.

The Program Design Step of the OFP Design and Mechanization Phase should begin with multi-level DFD definition of logical processes required to transform inputs into the desired outputs. All terms relating to data flows, data bases, and transforms of the logical design should be precisely documented


Algorithm
1 Define Logical Processes In Multi-level DFDs
Definition_of_ Dataflows_ Databases_ Processes
2 Create Data Dictionary
System_ Data_ Dictionary
Complete_Set_ Of_DFDs_For_ Program
4 Create Hierachical Modular System
Defined_Worker_ Modules
Defined_Executive_ Modules
Structure_Charts
3 Precisely Define Inputs/ Outputs/Module Functions
Defined_ Modules
5 Determine System Interfaces
8 Anticipate Serious Design Problem Areas
9 Determine User's Priorities
System_Interfaces
Possible_Problem_Areas
User's_Priorities
7 Create Detailed Specs For Modules
Program_Design
Detailed_Design
6 Verify OPP Design
Deficient_ Design
Verified_Detailed_ OPP_Design

Program Design Strategies	Program Design Tools	
	Hardware	Software
Top Down Design	General Purpose Processor	Data Base Management
Transform Analysis	Software Test Bed	System
Transaction Analysis	Interpretive Computer	Auto Documentation
Coupling Analysis	Simulation	Tools
Cohesion Analysis	Diagnostic Emulation	Design Tools
Morphology		
Jackson Design Strategy		
Warnier-Orr Strategy		

Figure 28. Design Program

in a data dictionary. The next step involves the creation of hierarchical modular systems from the processes and data flows specified in the data flow diagrams. At the top of the hierarchy are executive modules making decisions concerning global matters. Subordinate to the executive modules are worker modules dealing with details of the high level decisions. This process should be documented in a structure chart. In developing the structure chart, each module will be viewed as a black box. Module functions, inputs, and outputs should be defined but definitions of how each module will accomplish its functions should be deferred until later in the program design process. Modules should be broken down until the coding required will be manageably small and relatively independent. Once the black-box, modular design is complete, major interfaces in the system should be considered as well as potentially serious design problem areas and user's priorities in system implementation. With these considerations in mind, the designers will begin creation of detailed specifications for the system modules. When the detailed specifications for each module are complete and all interfaces, inputs, and outputs are defined, the Program Design Step of the Design and Mechanization Phase is complete. Figure 28 illustrates this process and lists some pertinent program design strategies. References 34 and 30 contain an abundance of information on these design strategies.

During the Detailed Program Design process, the General Purpose Processor and the STB are used. The DBMS running on the General Purpose Processor tracks the OFP status and the

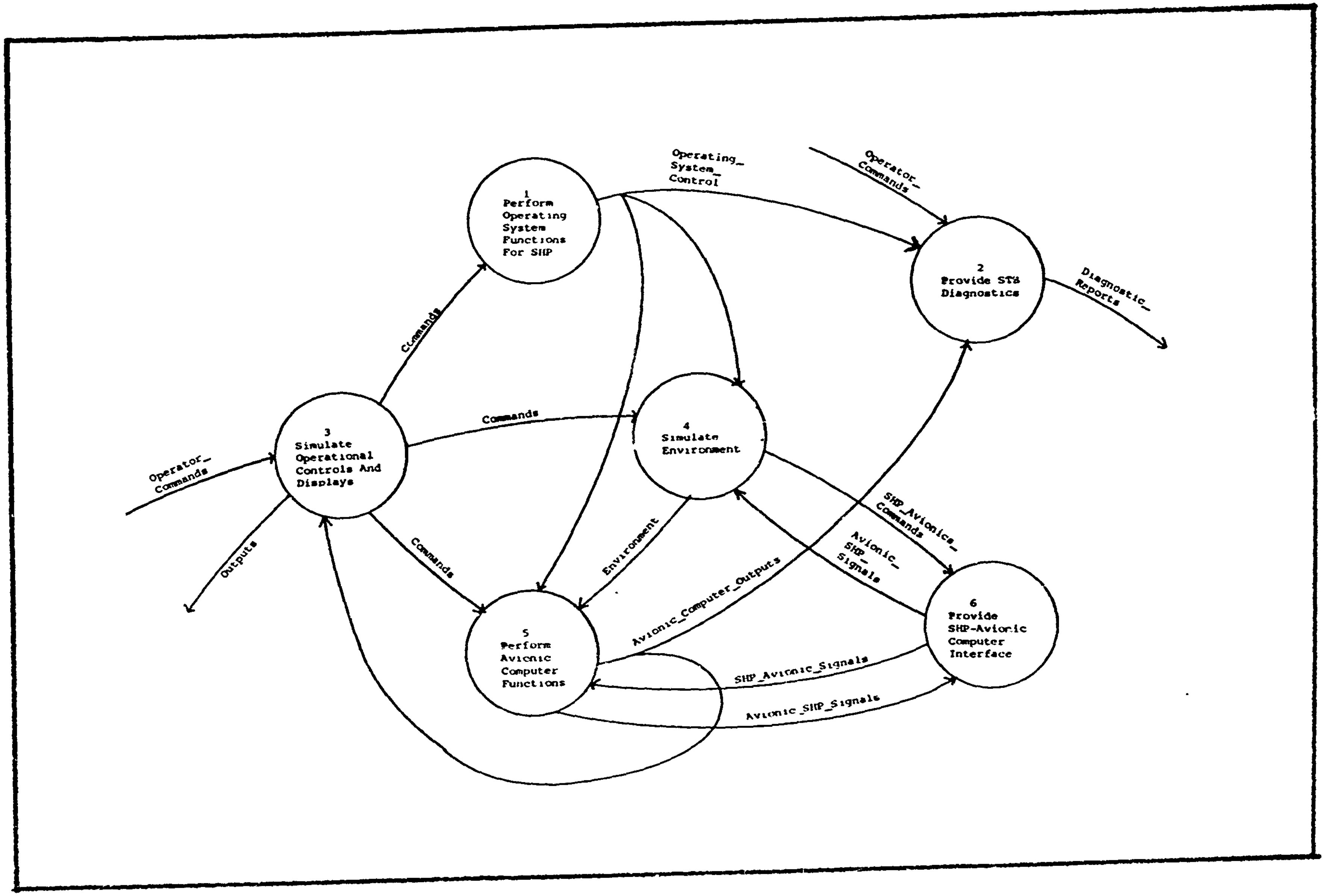


Figure 29. Verify OFP Design-The STB Functions

AISF configuration. Automated documentation tools, hosted on the General Purpose Processor are used to accomplish such actions as the drawing of Data Flow Diagrams and structure charts. The DBMS should also be used to hold the system data dictionary.

The OFP developer requires the capability of exercising proposed algorithms under simulated operational situations. Final selection of logic and program structure is determined by running trial cases using the Software Test Bed (STB). This tool allows real time software to be verified by interfacing the avionics computer with another computer, the Simulation Host Processor (SHP), which provides the input and environmental simulation. This process is represented by the bubble labeled Verify OFP Design in Figure 28. The functions required to accomplish this process are detailed in Figure 29.

In the STB, the operational controls, displays, and environment are simulated on the SHP in the form of mathematical models or appropriate software. The type processor best suited for the SHP is the low-cost mini or midi computer. Their low cost permits utilization on a dedicated basis for real time simulation applications. In selecting the SHP a comparative analysis should be made weighing such factors as word size, throughput, memory address capability, networking possibilities, initial cost, the accompanying support software, and whether other computers at the facility are from a software-compatible family. (Ref 5:32)

The SHP used in the STB will require an accompanying operating system which will provide for adaptation of the

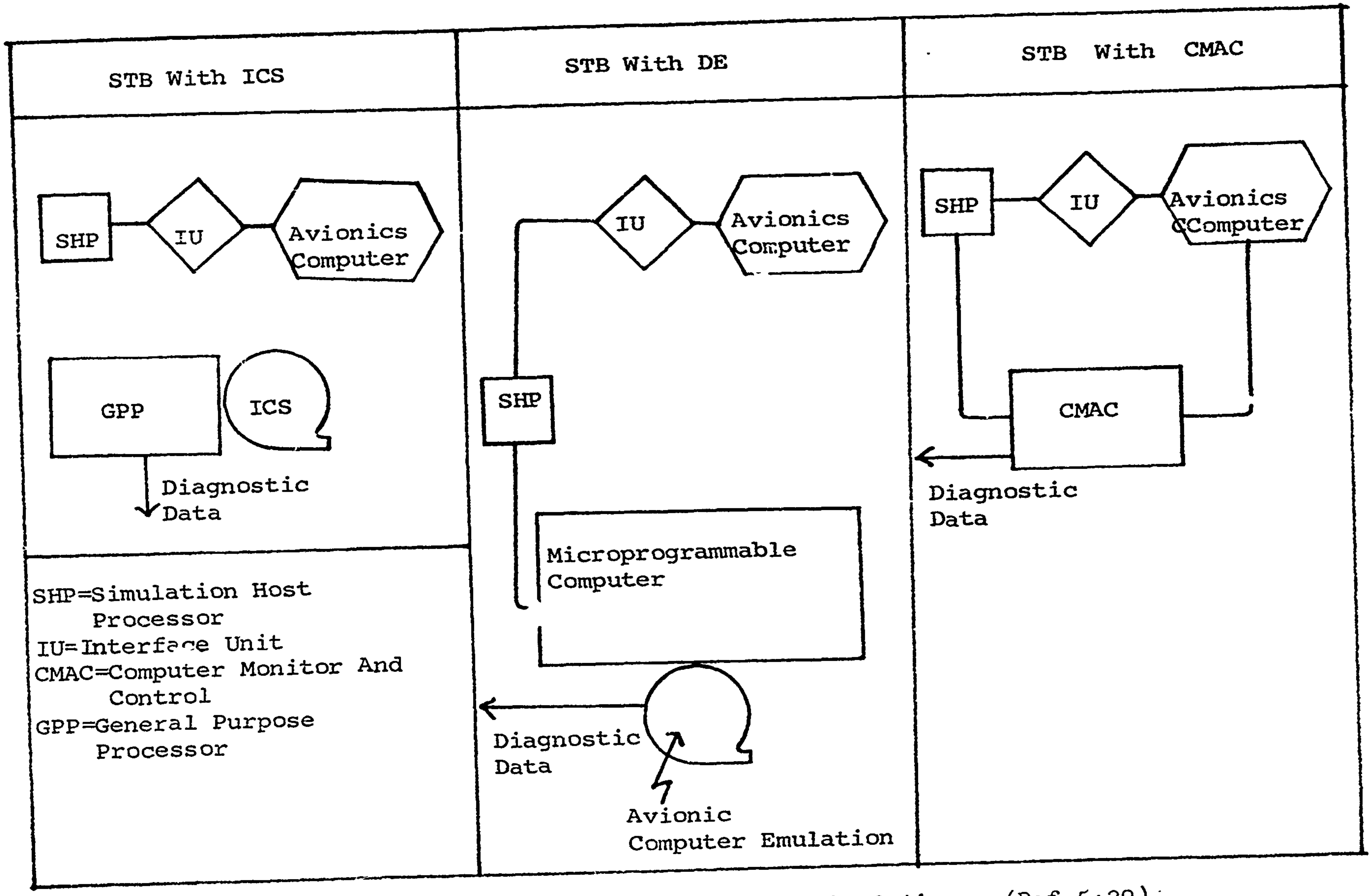

STB With ICS
STB With DE
STB With CMAC
SHP
IU
Avionics Computer
GPP
ICS
Diagnostic Data
Microprogrammable Computer
Avionic Computer Emulation
CMAC
SHP=Simulation Host Processor
IU=Interfa^e Unit
CMAC=Computer Monitor And Control
GPP=General Purpose Processor

Figure 30. STB With Diagnostic Options (Ref 5:29)

STB application software to the SHP hardware and for utilization of system resources.

The avionic computer must be procured and dedicated to the AISF and computer-to-computer hardware interfaces provided to link the avionic computer to the SHP. The OFP is resident on the avionic computer. Avionics hardware should be minimized in order to maintain configuration simplicity.

The last functional requirement for the STB is a provision for test bed diagnostics. The choice of a method for collecting operational software data for diagnostic purposes is an important consideration in the development of an AISF. The three alternatives, illustrated in Figure 30, are Interpretive Computer Simulation (ICS), Diagnostic Emulation (DE), and Computer Monitor and Control (CMAC).

An ICS is a program that performs the simulation of the avionic computer on a host computer which is generally the General Purpose Processor. This simulation is accomplished by generating an equivalent subroutine on the host computer for every operation code of the target computer. Memory locations in the host computer are reserved to represent the memory and registers of the avionics computer.

A DE provides essentially the same functions as an ICS, but it is implemented on a medium scale microprogrammable computer rather than a large scale General Purpose Processor. On a DE, the simulation of the avionic computer is written microcode instead of assembly language or an ICS language. The advantage of a DE is that simulation speed is faster at the microcode level with a potential speed advantage of 10

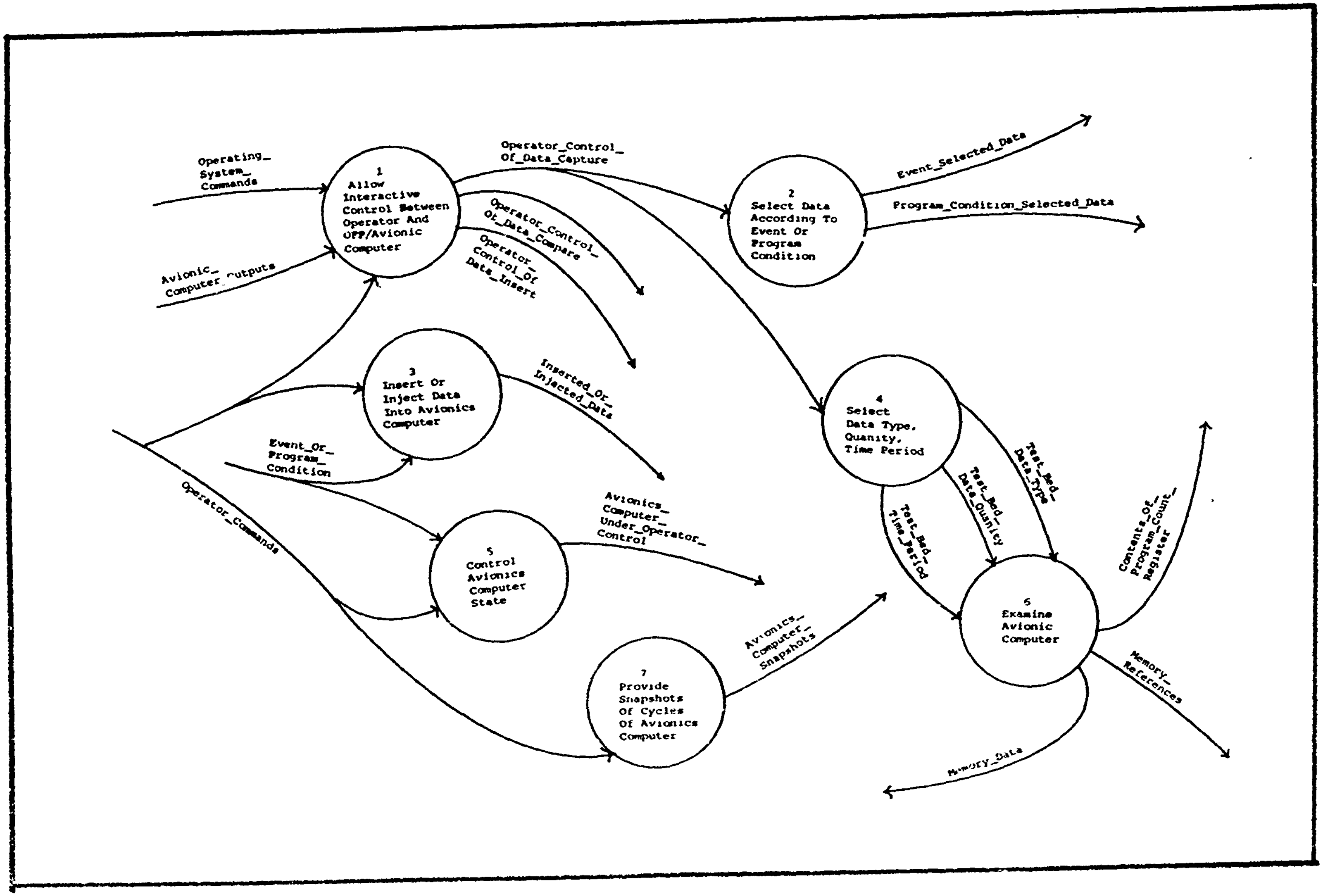


Figure 31. Functional Requirements For CMAC

times over the ICS.

The most direct method of providing diagnostic capability for a STB is the Computer Monitor and Control (CMAC). The design of a CMAC and its diagnostic capability depend somewhat upon the design of the avionic computer. The type of data which can be collected depends upon the accessability of the avionic computer's internal registers, memory and address register, program counter etc... through the Aerospace Ground Equipment (AGE). The intent of the CMAC is to allow the test operator to set breakpoints, loop, monitor, trace, and otherwise test the OFP in its hosted environment in real time without any special test-oriented code within the OFP. Figure 31 details the functions to be performed by the CMAC in the provision of STB diagnostics.

The CMAC must allow interactive control between the STB operator and the avionics computer/OFP in a near real time situation. This control will allow operator control of data capture, data compare, and data insert from the operator's console to the target processor. These capabilities allow the operator to respond to changing situations immediately without time-consuming rearranging of patch panels or reprogramming. Captured data should be selectable by the operator according to event and/or program conditions. Prior to accomplishing a test run the operator should be able to select the data type, quantity, or time period and examine the program count register, memory references, and memory data. Upon event or certain program conditions, the operator should be able to insert or inject data and control the

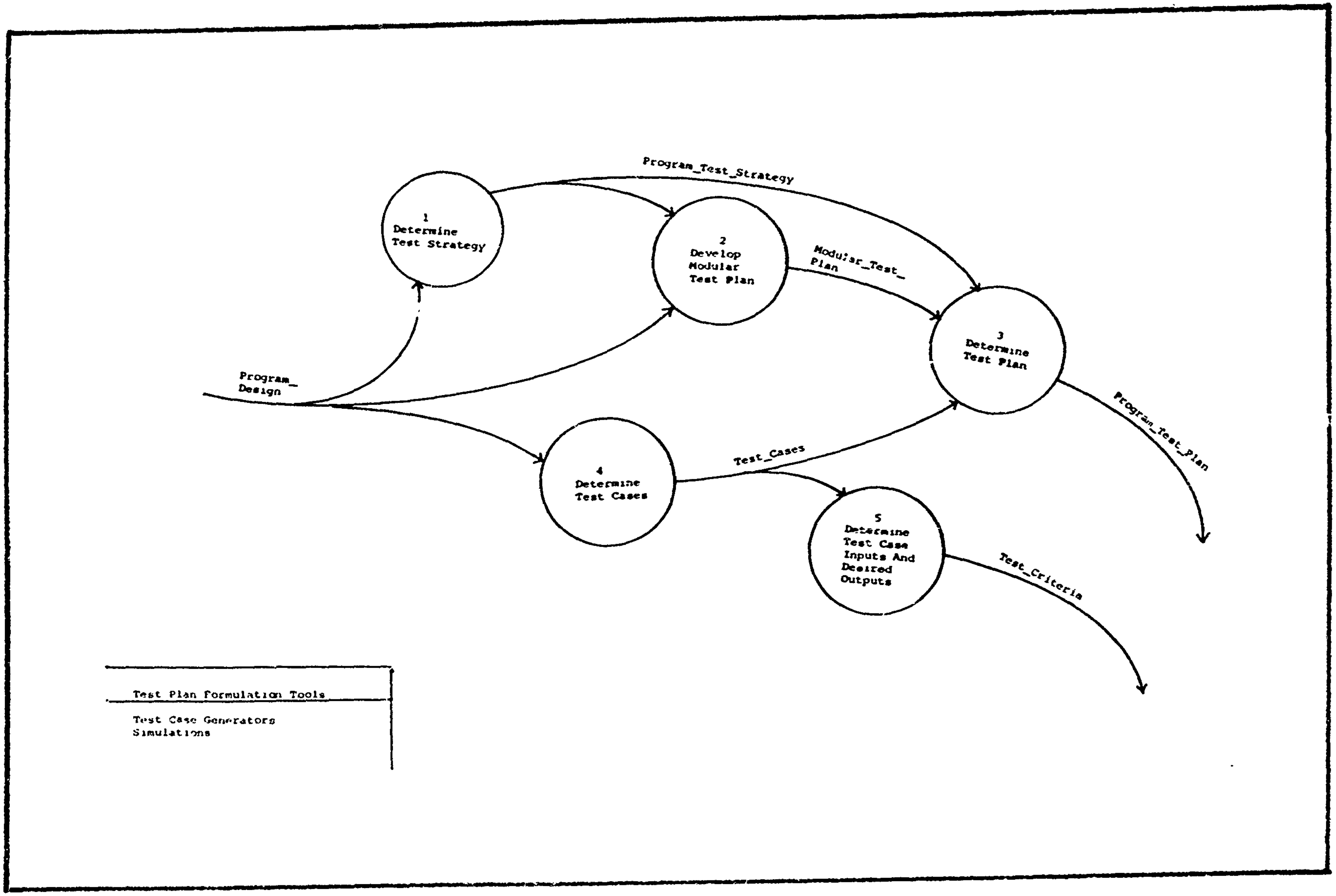


Figure 32. Develop Program Test Plan

state of the avionics computer.

Use of the CMAC should require no special or unique software in the target processor OFP. The CMAC should also be capable of capturing snapshots of cycles of the avionics computer, where a cycle consists of the processor events between execution of consecutive instructions.

Since the trend in computer design is toward standard internal bus structures (MIL-STD-1553), the enviromnent is being provided in which a nearly universal CMAC will be attainable.

The last step on the OFP Design and Mechanization Phase is the development of a program test plan. The program design produced in the previous step serves as an input to the process and desired outputs are a test plan and test criteria to ensure program viability. This process is illustrated in Figure 32. Initally, an overall program test plan strategy must be formulated. Top-down testing will eliminate the need to create driver modules in order to test lower level modules. The top level modules are implemented first and stubs coded to represent the lower level modules. This testing strategy will expose major interface problems before they affect the logic of lower level modules. Tests should be designed so that each successive test retests already accepted logic and each version of the system will provide the users with more functional capability, and logic accepted still will be valid in later versions. Incremental testing should also be a part of the program testing strategy. Once an overall program and modular test strategy is developed

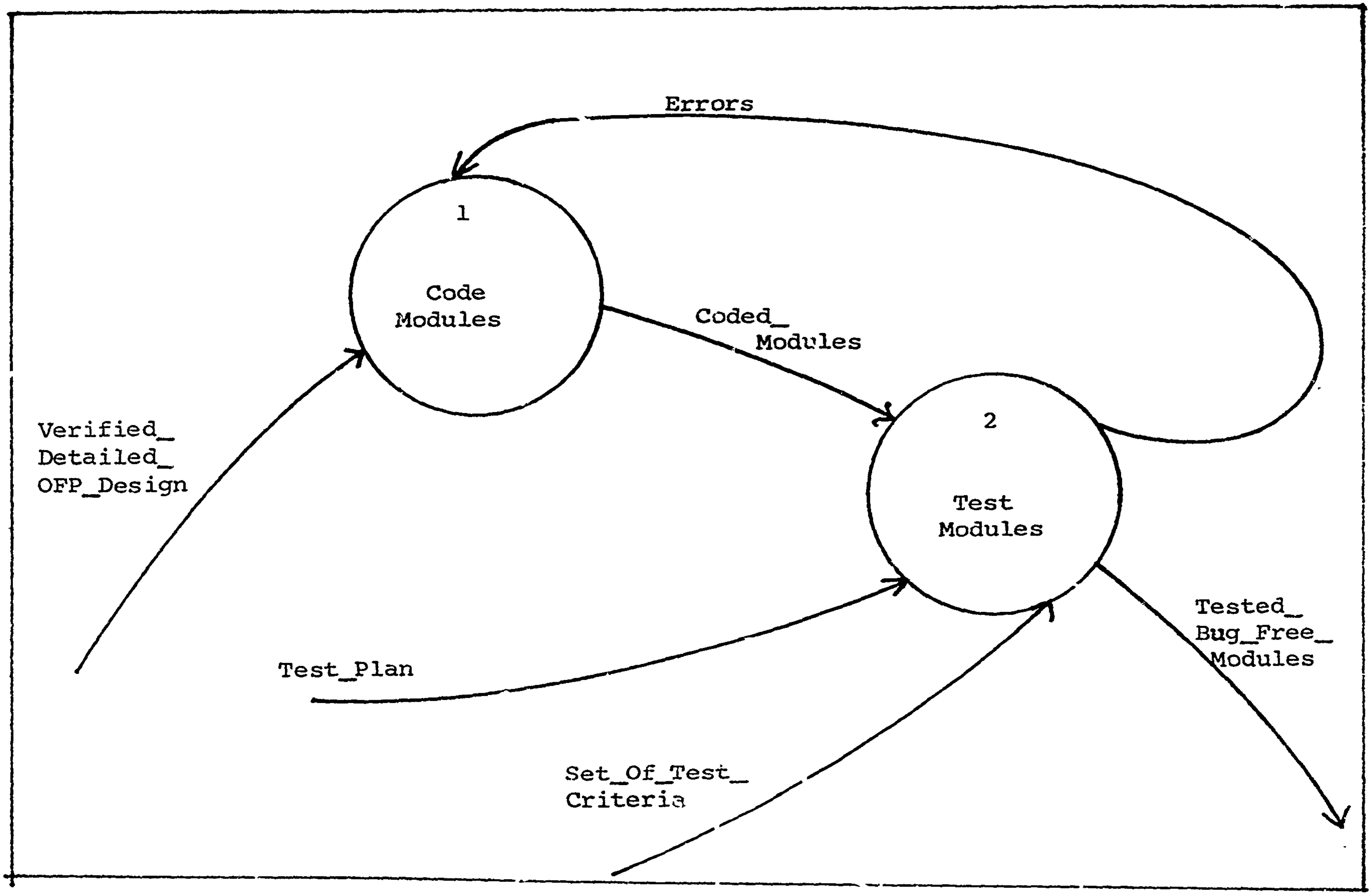
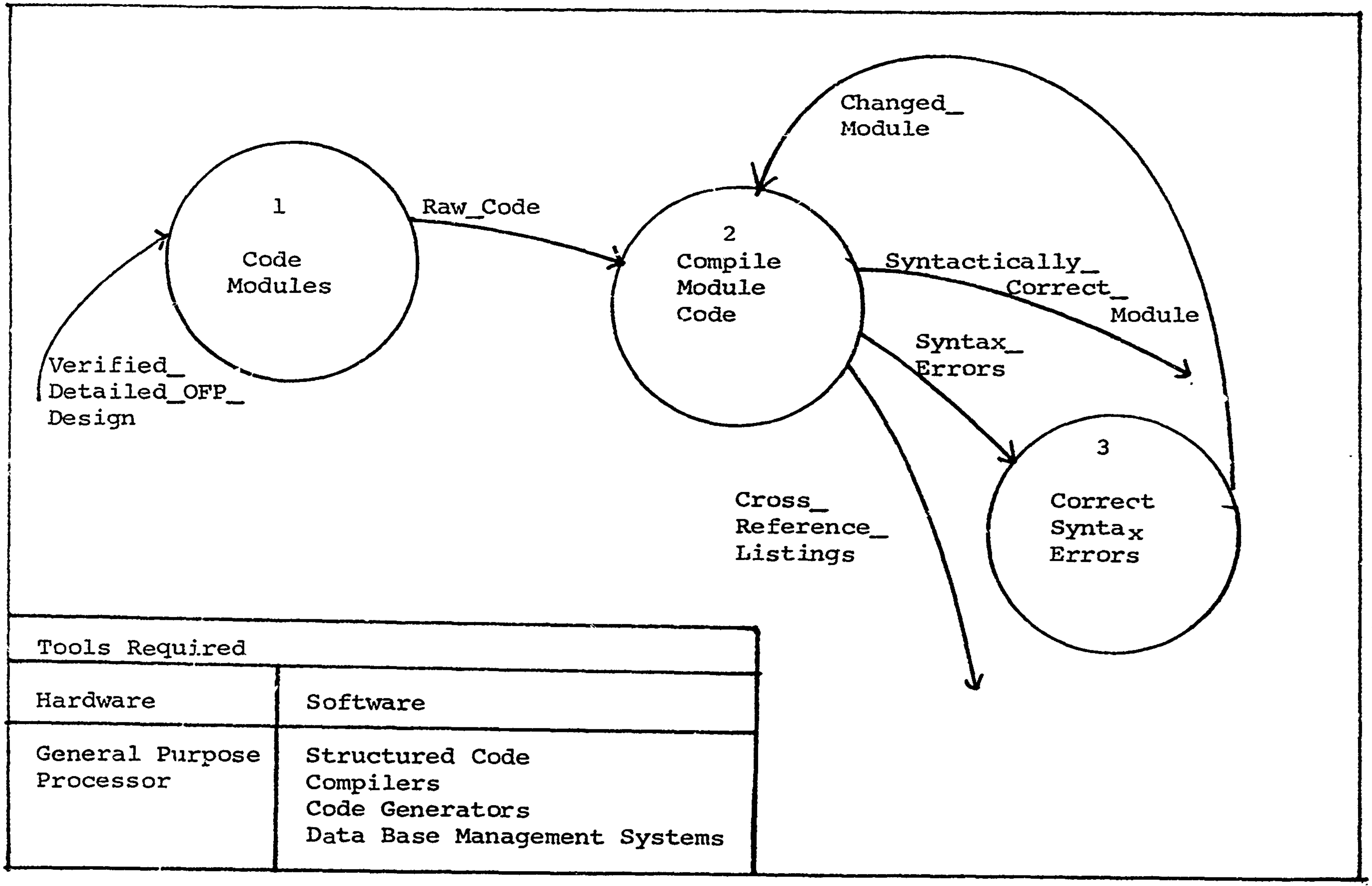


Figure 33. Code And Unit Test Functions



Tools Required	
Hardware	Software
General Purpose Processor	Structured Code Compilers Code Generators Data Base Management Systems

Figure 34. Code Functions

test cases must be generated. Test case inputs must be determined and desired outputs specified.

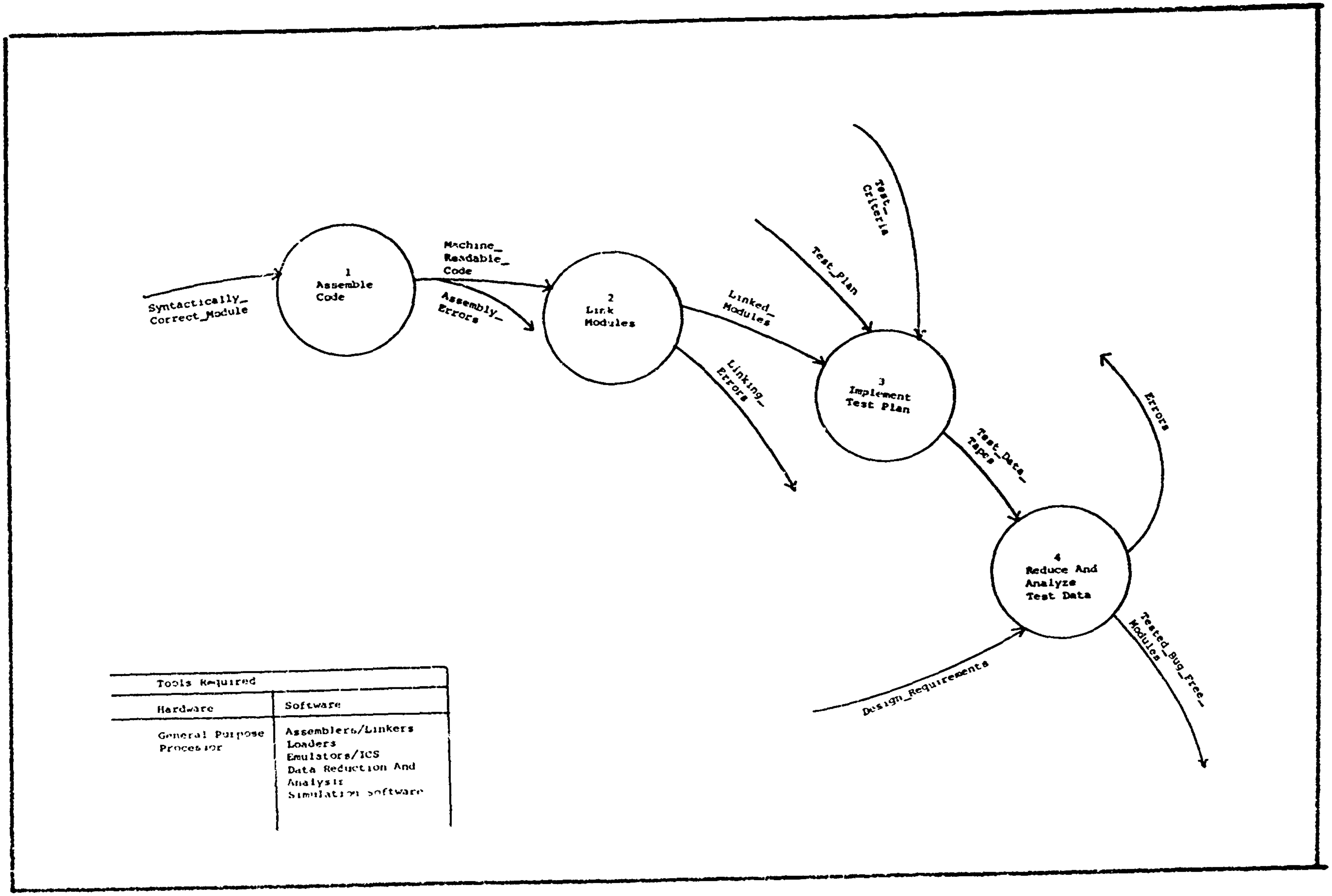
Following successful completion of the program detailed design and formulation of the program test plan, the next job to be accomplished at the AISF is that of coding and debugging the OFP.

Code and Debug Phase. Given a verified, detailed OFP design, the next step in the OFP change process involves coding and testing the modules comprising the design. A high level DFD illustrating the basic functions accomplished during this phase is provided in Figure 33.

The process of coding the OFP, shown in Figure 34, begins with the actual writing of code in the appropriate language. The code should reflect the modular structure produced in the Design and Mechanization Phase. Once raw code is produced it is then compiled to reveal syntax and language-peculiar errors. The syntax errors must then be corrected and the code recompiled. When the module compiles successfully it is ready for testing.

The major hardware resource employed to carry out the functions of the coding process is the General Purpose Processor. The code is written using this computer and compiled using a compiler resident on it. The DBMS also resident on the General Purpose Processor may maintain a data base containing source code produced during the process.

Following successful coding and compilation of the modules, the next function to be accomplished during this phase is the unit test of each module. This process is depicted in



Tools Required	
Hardware	Software
General Purpose Processor	Assemblers/Linkers Loaders Emulators/ICS Data Reduction And Analysis Simulation Software

Figure 35. Test Functions

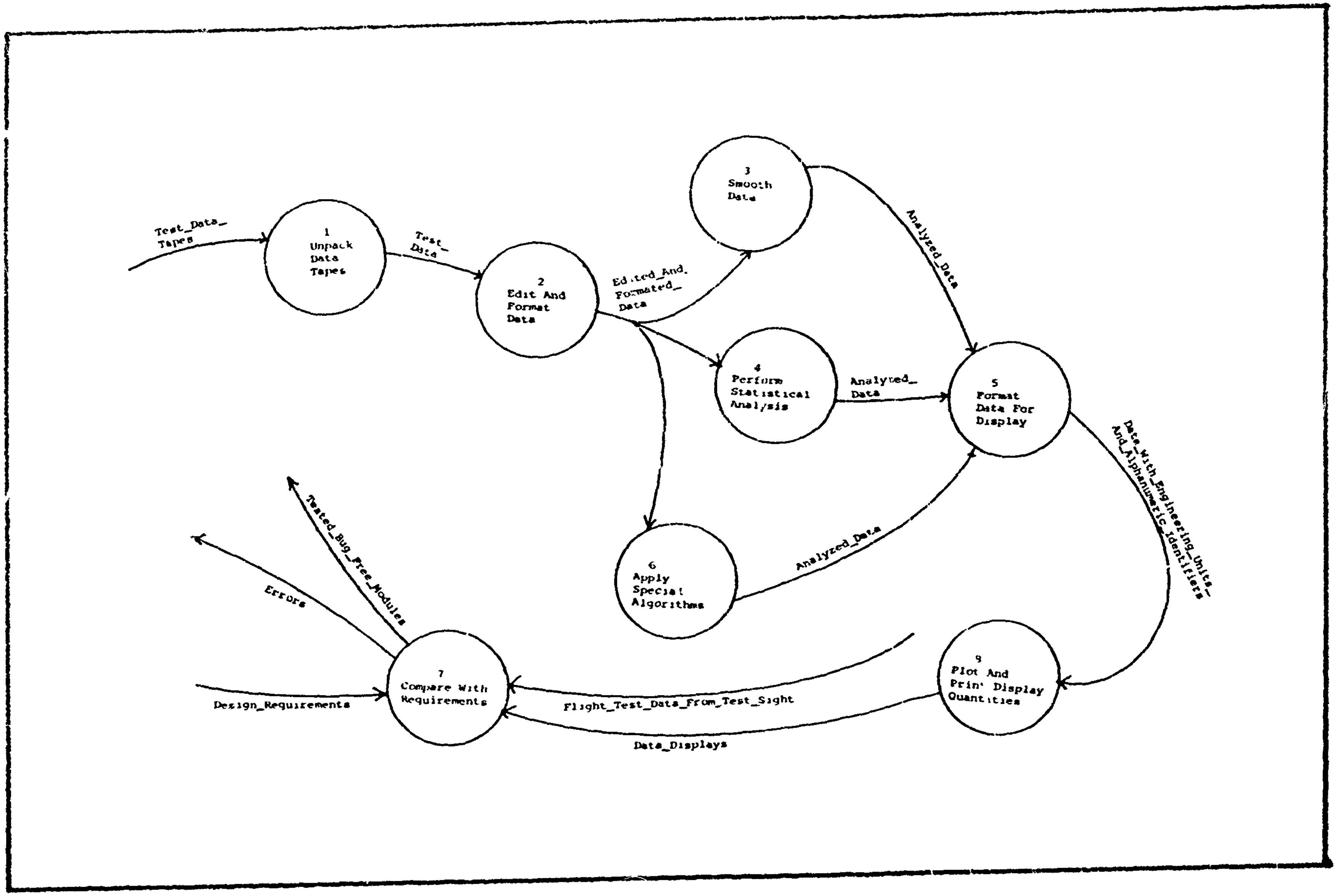


Figure 36. Data Reduction And Analysis Functions

Figure 35. The first functions to be accomplished during the Unit Test portion of the Code and Unit Test Phase are assembling and linking the modules. As each module is coded and compiled it is assembled then linked to the already existing baseline program. These processes may produce semantic errors which will have to be corrected through recoding. The successfully linked modules are then subjected to the test plan formulated during the Design Phase. The intent is to test the last module produced. Implementation of the test plan will yield quantities of raw data which must then be reduced and analyzed for compliance with the design and system requirements determined in previous phases.

During the Unit Test process, the General Purpose Processor hosts the assemblers and linkers. This processor also hosts the DBMS which should maintain data bases consisting of source and object code,and test cases and test results. The General Purpose Processor will also serve as host to the various Data Reduction and Analysis programs required to process raw test data.

The Data Reduction and Analysis software will be used to analyze test results from the simulations, test beds, and eventual flight tests. This software package will consist of a set of small programs designed to convert raw test data recordings into an organized, easily-interpreted set of plots and data listings. Functions performed during the Data Reduction and Analysis process are depicted in Figure 36. The specifics involved in accomplishing these functions are largely dependent upon the system being tested and such

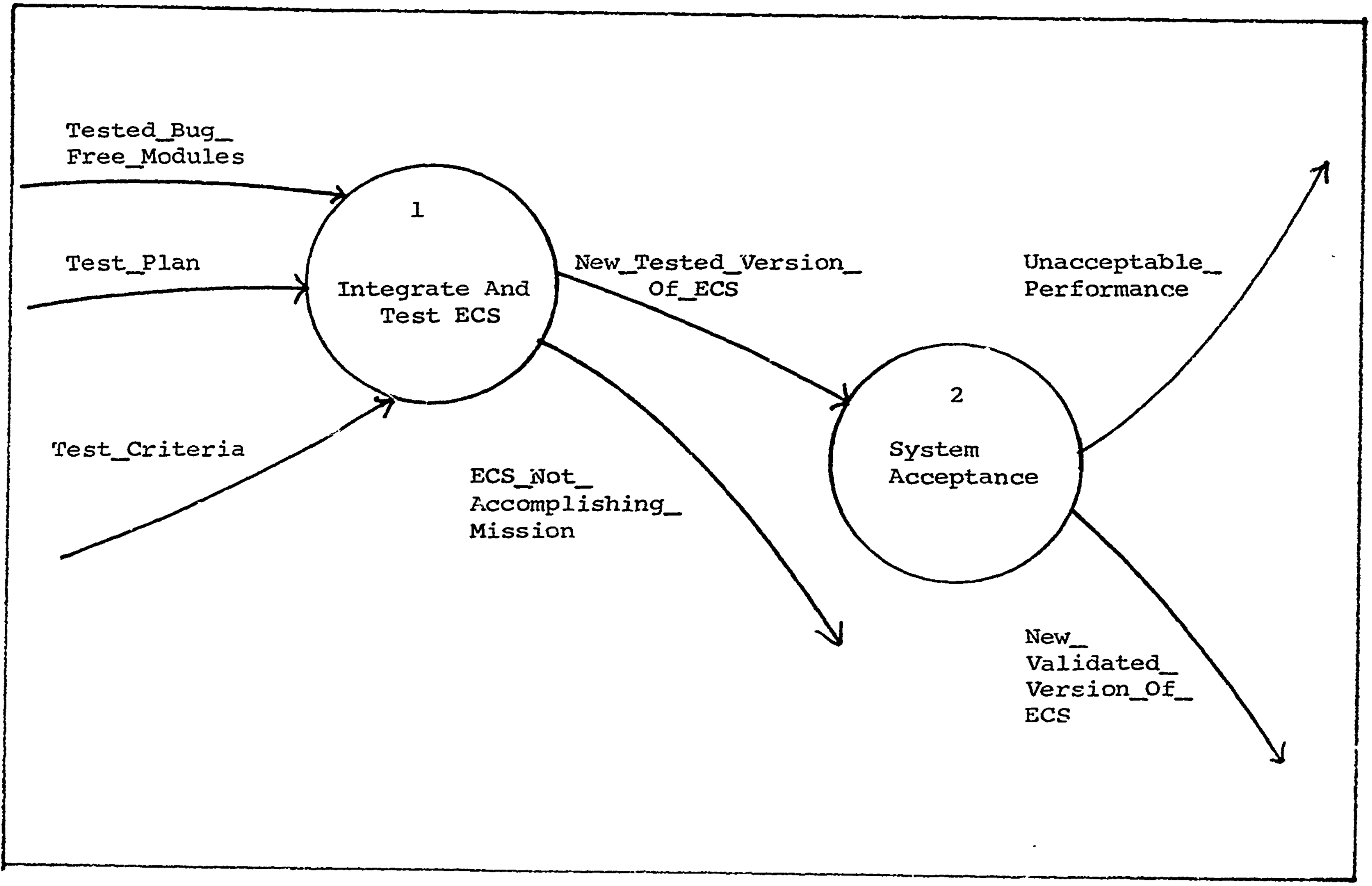


Figure 37. Testing And Verification Phase

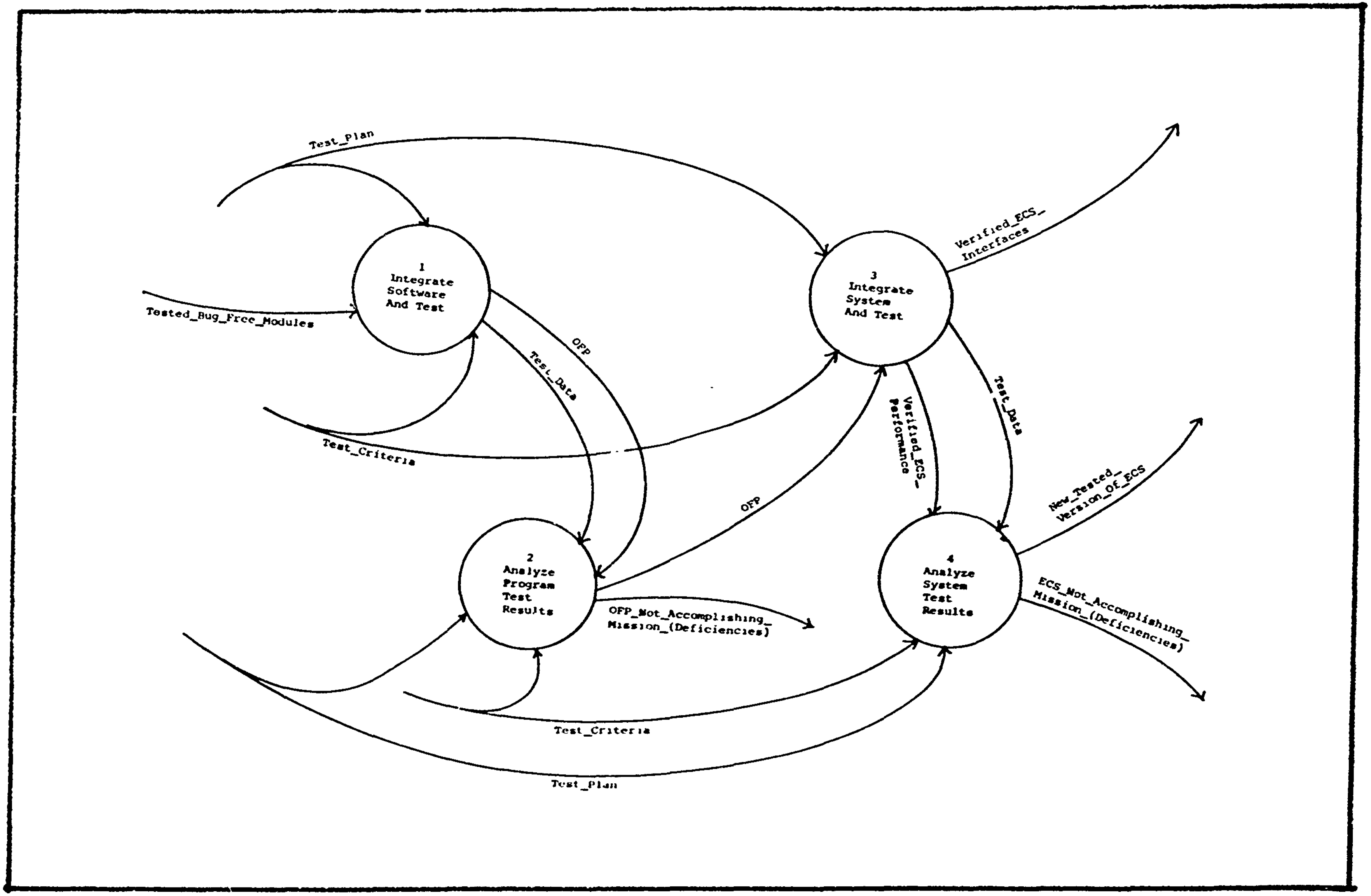


Figure 38. Integrate And Test ECS

software as the formatters and special algorithms for data analysis should be written especially for the project. Data tapes from the test beds should be formatted on computer-compatibl3 magnetic tape and flight test data,which will have already been processed at the test site,should also be on a computer-compatible tape. The Data Reduction and Analysis software should be designed to display variables from flight tests and test beds in the same manner for ease of comparison between the two modes of testing.

Testing and Verification Phase. The final phase in the OFP change process, the Testing and Verification Phase, actually encompasses two major steps: Integration, and, Test and System Acceptance. The Testing and Verification Phase receives the tested, bug-free modules from the Code and Debug Phase, along with the system test plan and test criteria from the Design and Mechanization Phase. Output from this process is the new, verified version of the Embedded Computer System (ECS). Figure 37 is a high level diagram of this phase. Functions performed during the Integration and Test portion of this phase are depicted in Figure 38.

The first major function performed is the integration and testing of the software (OFP). Output from this process consists of an integrated OFP and the raw test data generated during the OFP testing process. The raw test data is then reduced and analyzed to determine whether the OFP is complying with the requirements stipulated in the Analysis Phase. If test results indicate that the OFP is functioning properly it is then ready to be integrated into the complete ECS;

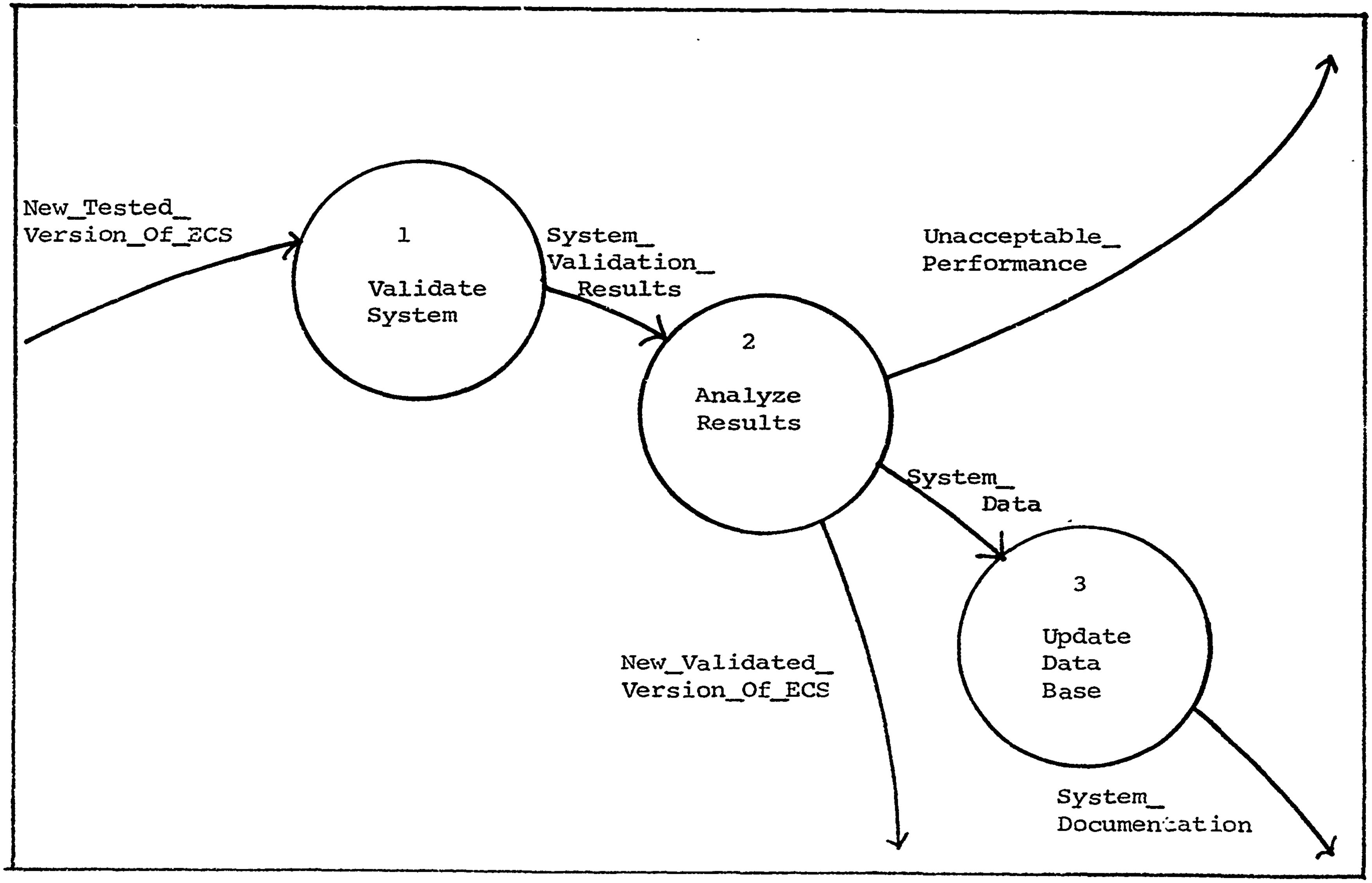
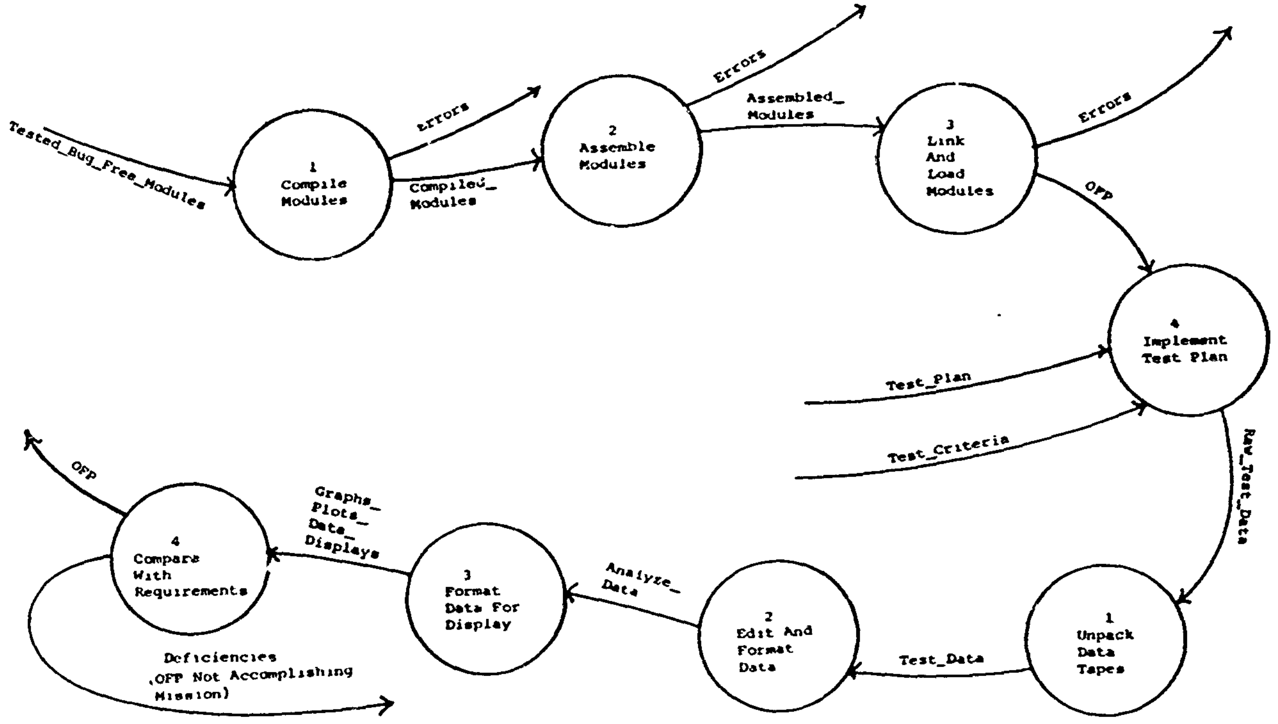


Figure 39. System Acceptance

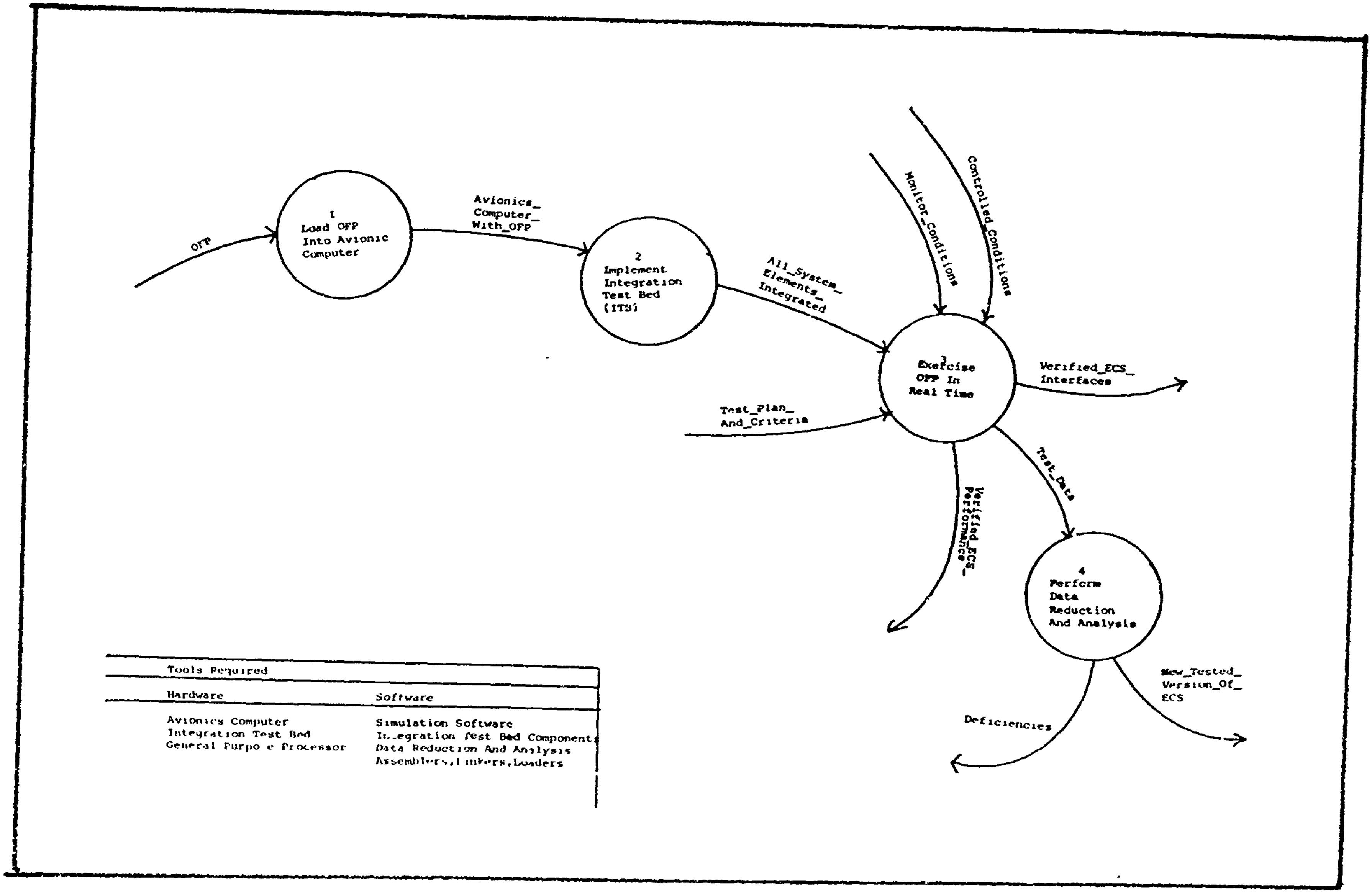


Tools Required	
Hardware	Software
General Purpose Processor Software Test Bed	Assemblers, Compilers Loaders Data Reduction and Analysis Simulation Software

Figure 40. Integrate Software And Test And Analyze Program Test Results

otherwise the change process must be reaccomplished in order to correct program deficiencies. ſhe system Integration and Test procedure involves integrating the new OFP and the ECS hardware to form a new ECS, and testing the system to generate raw test data. This test data must then be analyzed for compliance with system specifications. If the ECS complies with the system specifications it is then ready for the system acceptance process, otherwise the change process must be reaccomplished to correct system deficiencies. The functions composing the System Acceptance process are illustrated in Figure 39. Input to the System Acceptance process is the new, tested, version of the ECS from the Integrate and Test process. The first function to be accomplished is system validation. Results from this action must be analyzed for proper system performance. If the system is deemed acceptable at this stage, the new, validated version of the ECS is accepted. The last required function in the System Acceptance process is the updating of the system data base and system documentation.

The next series of figures reduces the functional requirements for the Testing and Verification Phase to the next level of detail and shows hardware and software tools required to accomplish the functions. Figure 40 depicts the functions of the Integration and Test of the software and analyzing the test results. The modules produced in the Code and Unit Test Phase are compiled, assembled, linked, and loaded. These processes are accomplished on the General Purpose Processor. Once the program has been linked and loaded it is



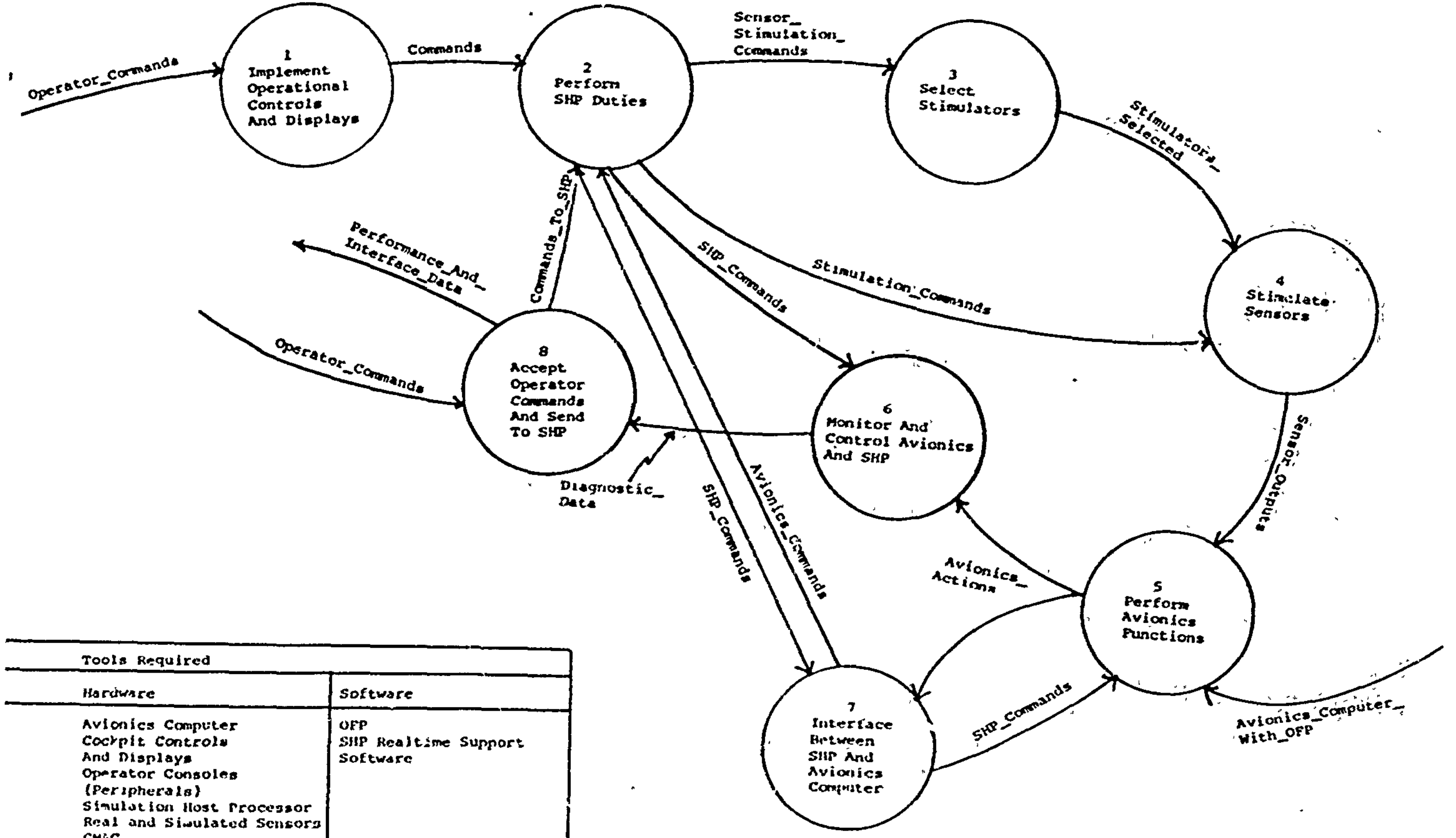
Tools Required	
Hardware	Software
Avionics Computer Integration Test Bed General Purpo e Processor	Simulation Software In.egration Test Bed Components Data Reduction And Analysis Assemblers,Linkers,Loaders

Figure 41. Integrate System And Test And Analyze System Test Results

ready for testing. The test plan developed during the Design and Mechanization Phase is implemented to produce raw test data. The primary tool used during this testing is the Software Test Bed (STB). STB functional requirements were described in the Design and Mechanization Phase section of this report. Following the testing of the OFP, the raw test data must be reduced and analyzed. Data Reduction and Analysis programs running on the General Purpose Processor will accomplish the tasks of unpacking the data tapes, editing and formatting the data, and producing data analysis reports. Findings from the Data Reduction and Analysis activity are compared with the requirements specified during the Analysis Phase. If requirements are satisfied the OFP is then prepared for system Integration and Test. Functional requirements and tools needed for this process are shown in Figure 41.

The OFP is first loaded into the avionics computer. The avionics computer and resident OFP comprise a major component of the Integration Test Bed (ITB). The remainder of the ITB is implemented and real time exercise of the OFP is accomplished. Through this process, the ITB is used to verify the ECS interfaces and performance. Raw test data is also produced. This data is processed in the manner previously discussed. The product of this process is the new, tested, version of the ECS ready for system acceptance.

The major hardware components required for the System Integration and Test are the General Purpose Processor for hosting the Data Reduction and Analysis programs and the linkers and loaders, the ITB for real time testing and integration


Operator_Commands
1 Implement Operational Controls And Displays
Commands
2 Perform SHP Duties
Sensor_ Stimulation_ Commands
3 Select Stimulators
Stimulators_ Selected
4 Stimulate Sensors
Stimulation_Commands
SHP_Commands
Sensor_Outputs
5 Perform Avionics Functions
Avionics_Computer_ With_OFP
Avionics_ Actions
6 Monitor And Control Avionics And SHP
SHP_Commands
7 Interface Between SHP And Avionics Computer
Avionics_Commands
SHP_Commands
Diagnostic_ Data
8 Accept Operator Commands And Send To SHP
Commands_To_SHP
Performance_And_ Interface_Data
Operator_Commands

Tools Required	
Hardware	Software
Avionics Computer Cockpit Controls And Displays Operator Consoles (Peripherals) Simulation Host Processor Real and Simulated Sensors CMAC Interface Unit	OFP SHP Realtime Support Software

Figure 42. Implement Integration Test Bed

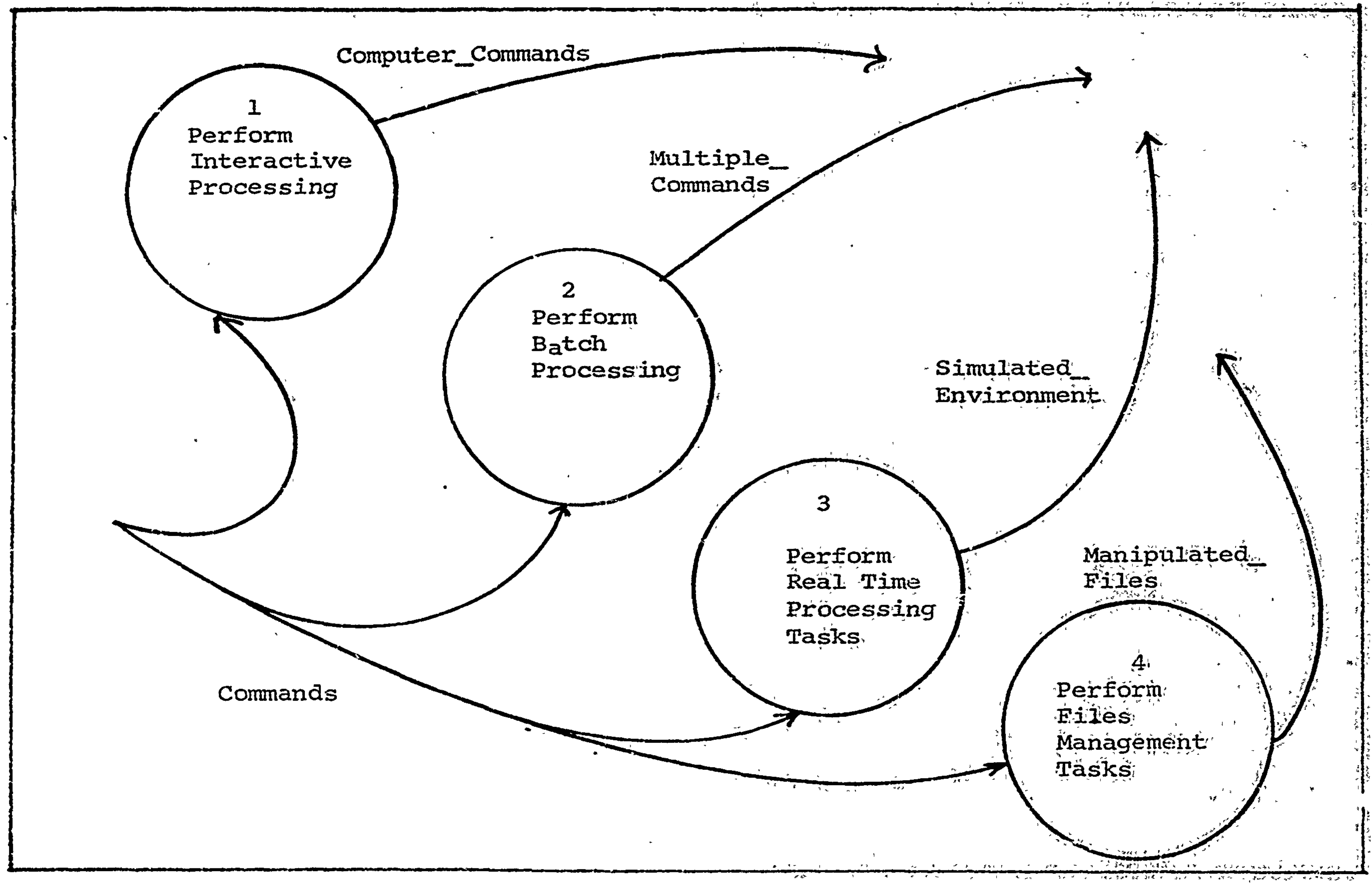


Figure 43. Perform SHP Duties

of the ECS, and the avionics computer hosting the OFP. Functional requirements and tools required for implementing the ITB are depicted in Figure 42.

The ITB emphasizes system integration testing by interfacing,within the test bed,selected sensors and other avionic equipment including the avionics computers and the actual cockpit controls and displays. The Simulation Host Processor (SHP) with interface hardware generates analog or digital signals providing stimulation to the sensors of the type normally received in flight. Sensors and stimulators will be the actual hardware from the aircraft or will be simulated. Operator commands are entered via the actual cockpit controls and displays. The SHP interfaces with the stimulators, sensors, CMAC, and the avionics computer through a hardware Interface Unit (IU). Other peripheral equipment may also be included for displaying and inputting data. Diagnostic capability should be provided by a CMAC of the type outlined in Figure 31. The SHP should be selected based upon a sizing and timing analysis of the real time simulation program. This analysis will provide an estimate of the memory address and throughput requirements for the SHP. Functional requirements for the SHP are shown in Figure 43. It must provide for interactive processing allowing multiple users to interact with the processor through CRT consoles and keyboards. Functions performed interactively include test definition and preparation, file manipulation, and batch job submission. The SHP will also be required to operate in a batch mode allowing it to perform large tasks not requiring immediate results such as

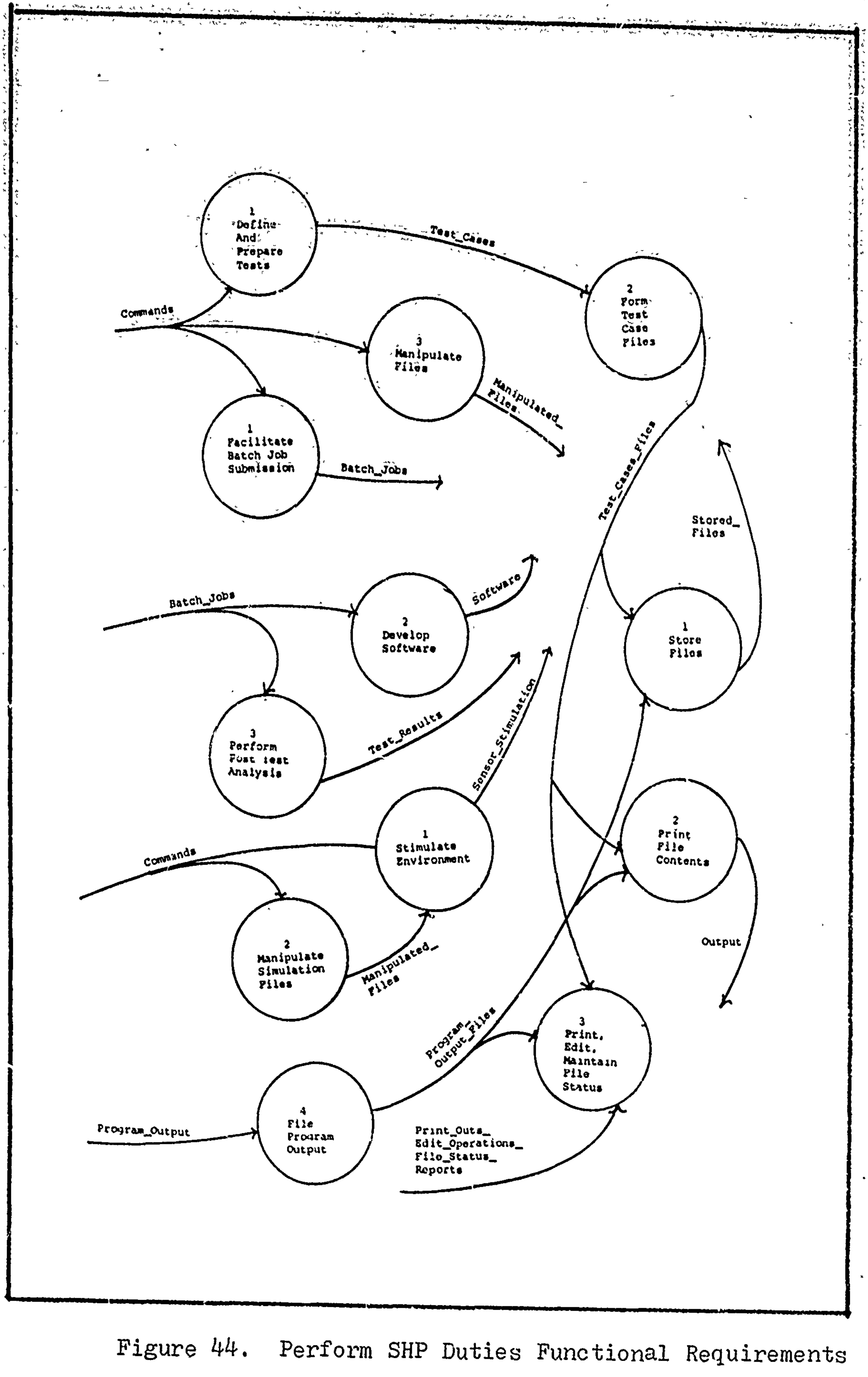


Figure 44. Perform SHP Duties Functional Requirements

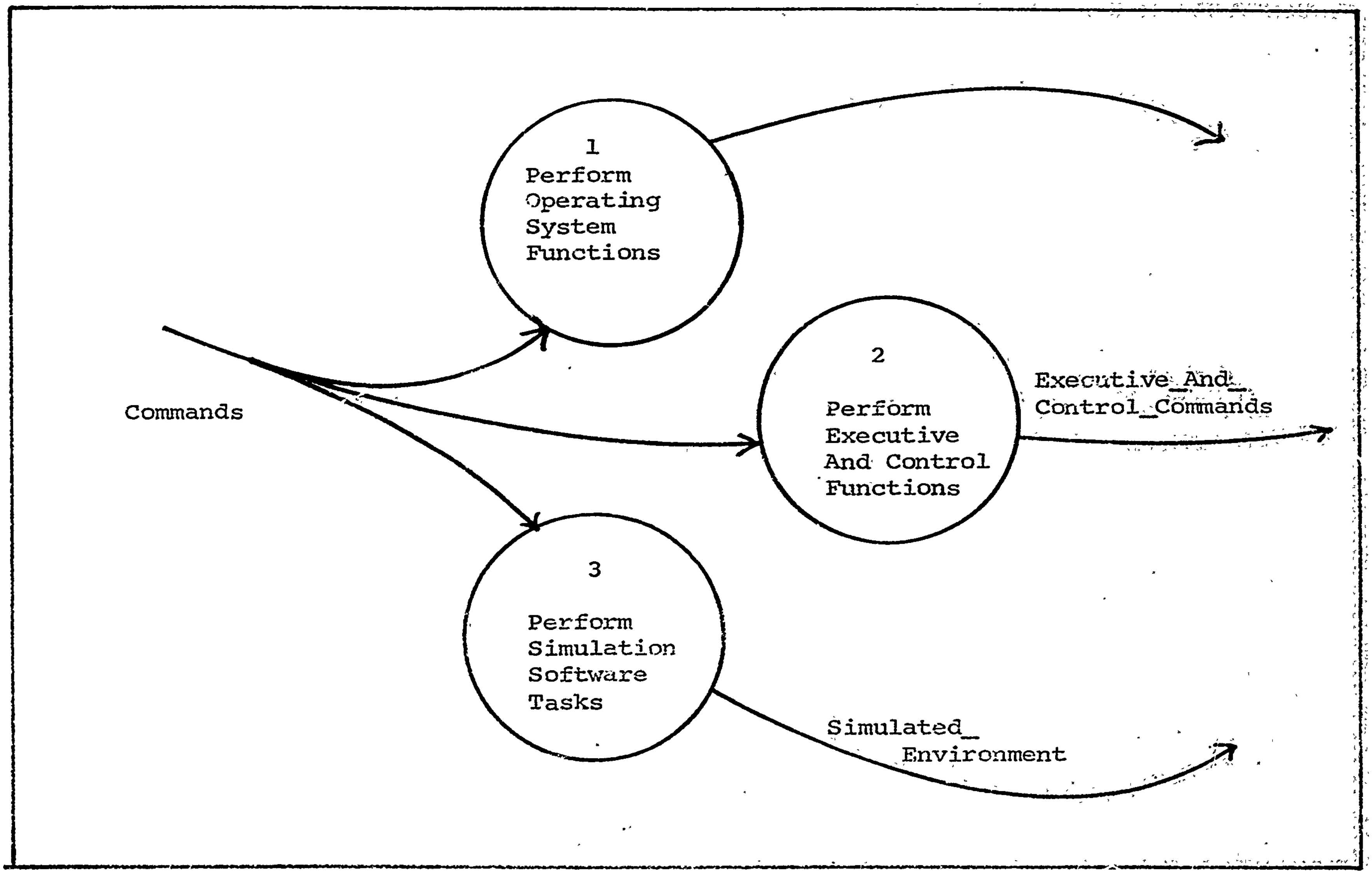


Figure 45. Real Time Support Software Functions

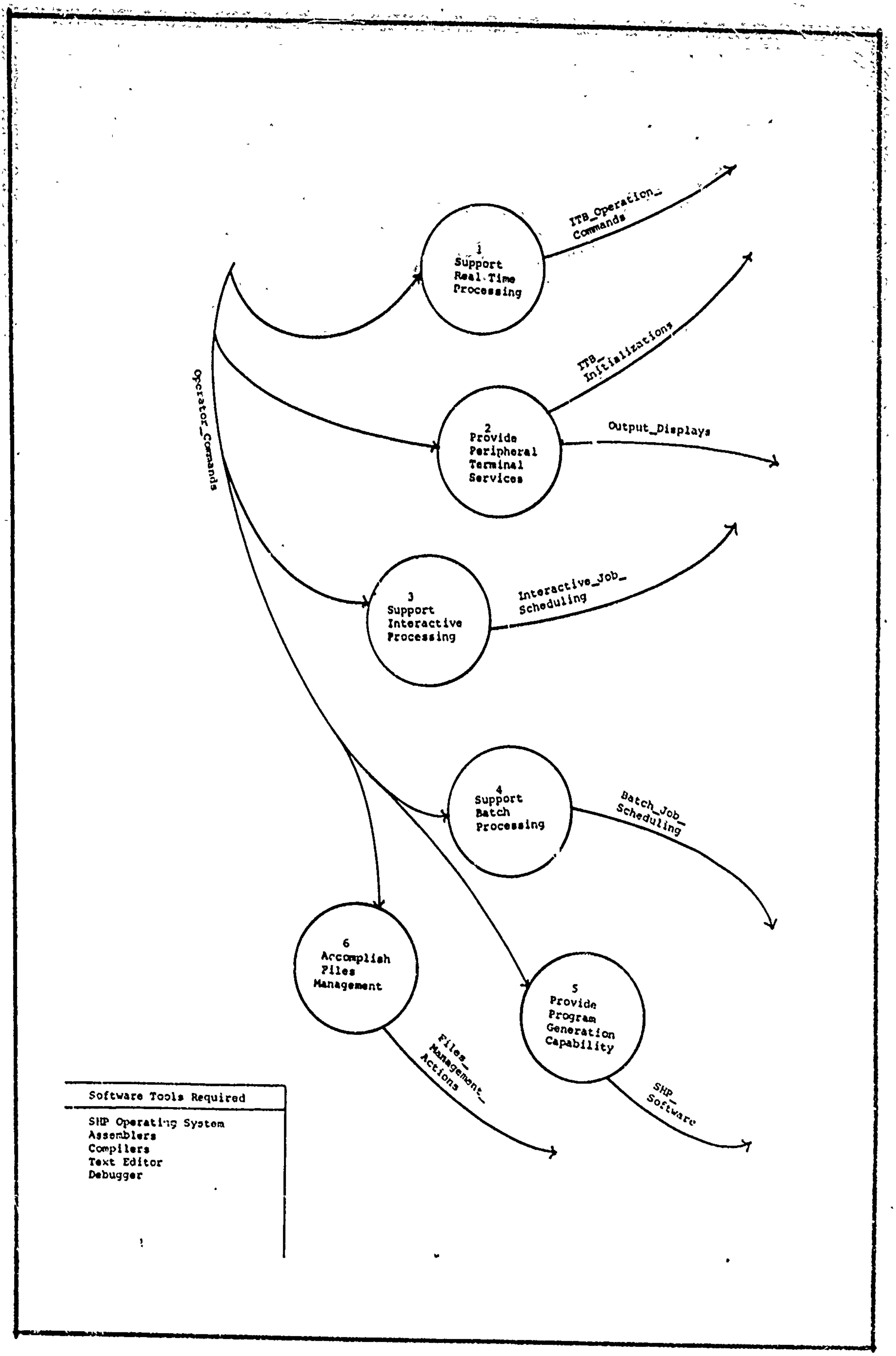


Figure 46. SHP Operating System Functions

SHP software development and post test analysis. The third functional requirement stipulates that the SHP must be capable of performing real time environmental simulation tasks on a shared basis with the routine tasks. Finally, the SHP must have the capability to perform files management tasks. Files, including source and object programs, different environmental simulations, program output, and test files are included. The SHP must also be capable of storing files on disk or tape and retrieval and printing of file contents. An edit capability and file status maintenance should also be provided. The functional requirements of Figure 43 are further detailed in Figure 44. If several STBs or ITBs are established using the same type SHP a distributed SHP may be considered. A single master SHP would perform the bulk of the functions delineated in Figure 44, while dedicated SHPs would be assigned to the various avionics subsystems.

In order for the SHP to perform its duties a complement of real time support software will be required. This software can be organized into the three major subdivisions: Operating System, Executive and Control Software, and Application Modules. The three main functions are illustrated in Figure 45. The next series of DFDs details these three functions and the real time support software required to implement them. Figure 46 shows the operating system functions. The functional requirements for the operating system will depend upon whether the SHP is strictly a dedicated test bed processor or a multi-functional processor with peripherals to support program generation of of SHP software. The functional requirements provided in

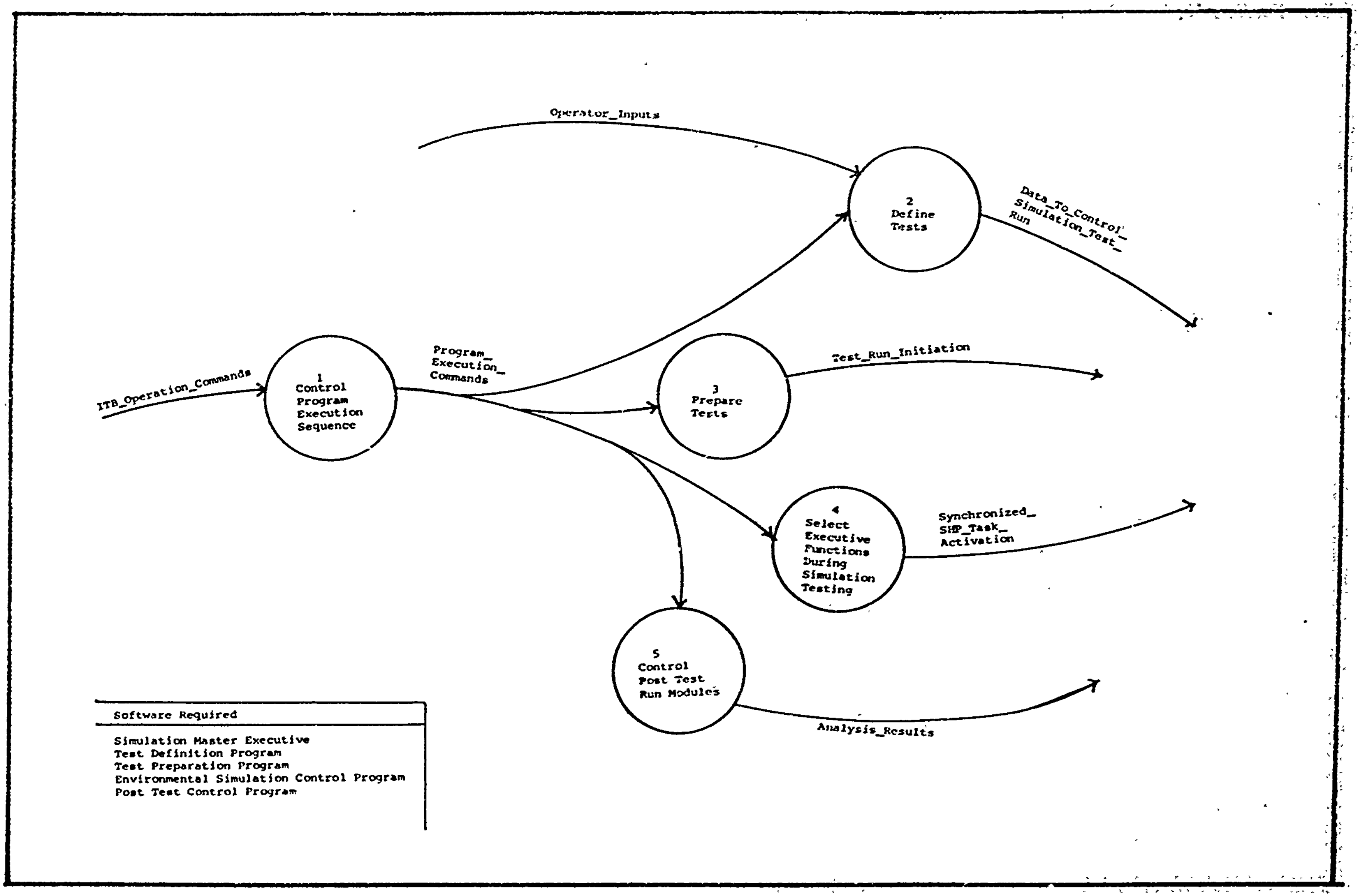

Operator_Inputs
1 Control Program Execution Sequence
2 Define Tests
3 Prepare Tests
4 Select Executive Functions During Simulation Testing
5 Control Post Test Run Modules
ITB_Operation_Commands
Program_ Execution_ Commands
Data_To_Control_ Simulation_Test_ Run
Test_Run_Initiation
Synchronized_ SHP_Task_ Activation
Analysis_Results
Software Required
Simulation Master Executive
Test Definition Program
Test Preparation Program
Environmental Simulation Control Program
Post Test Control Program

Figure 47. SHP Executive And Control Functions

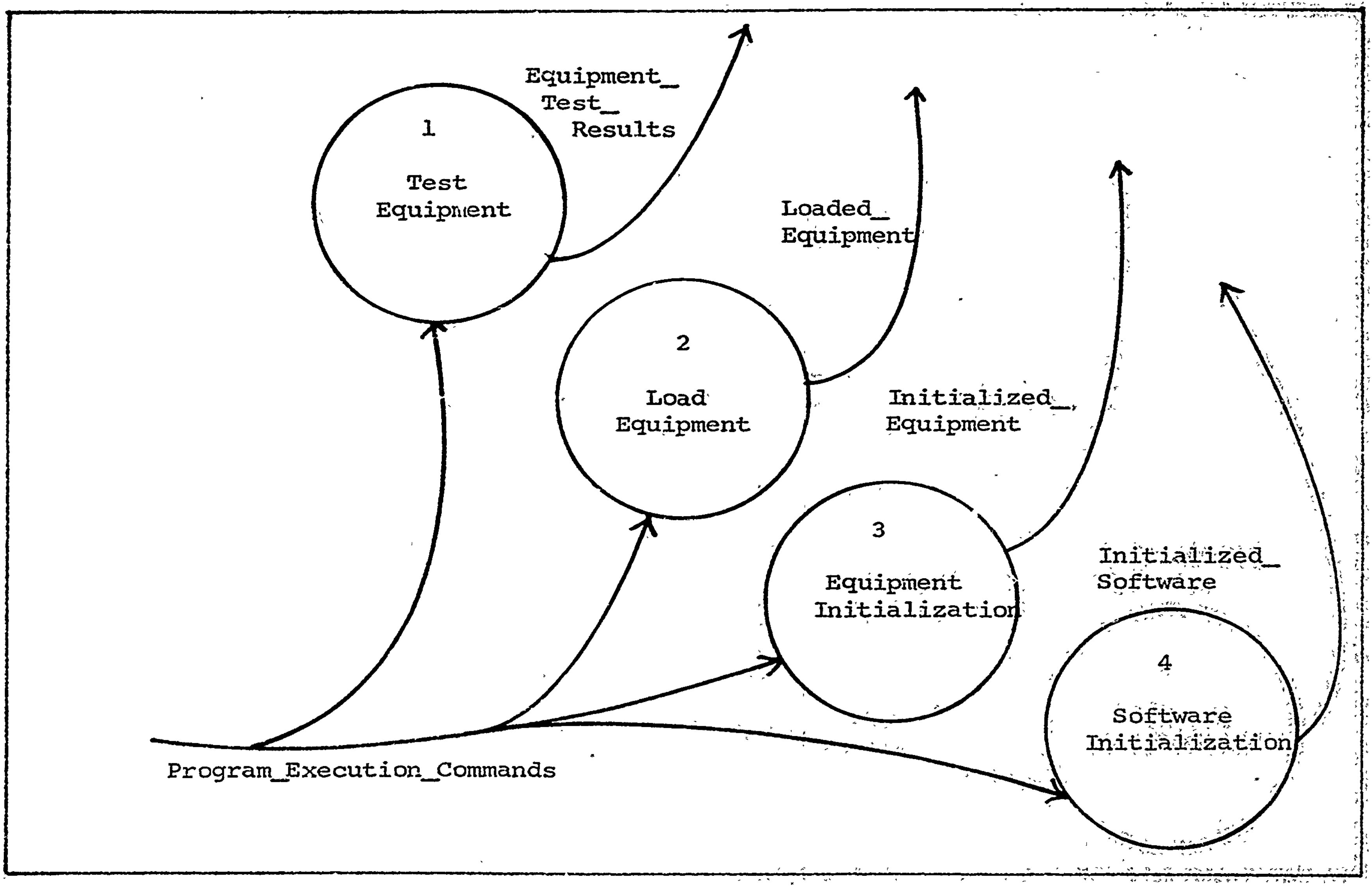


Figure 48. Prepare Test Functions

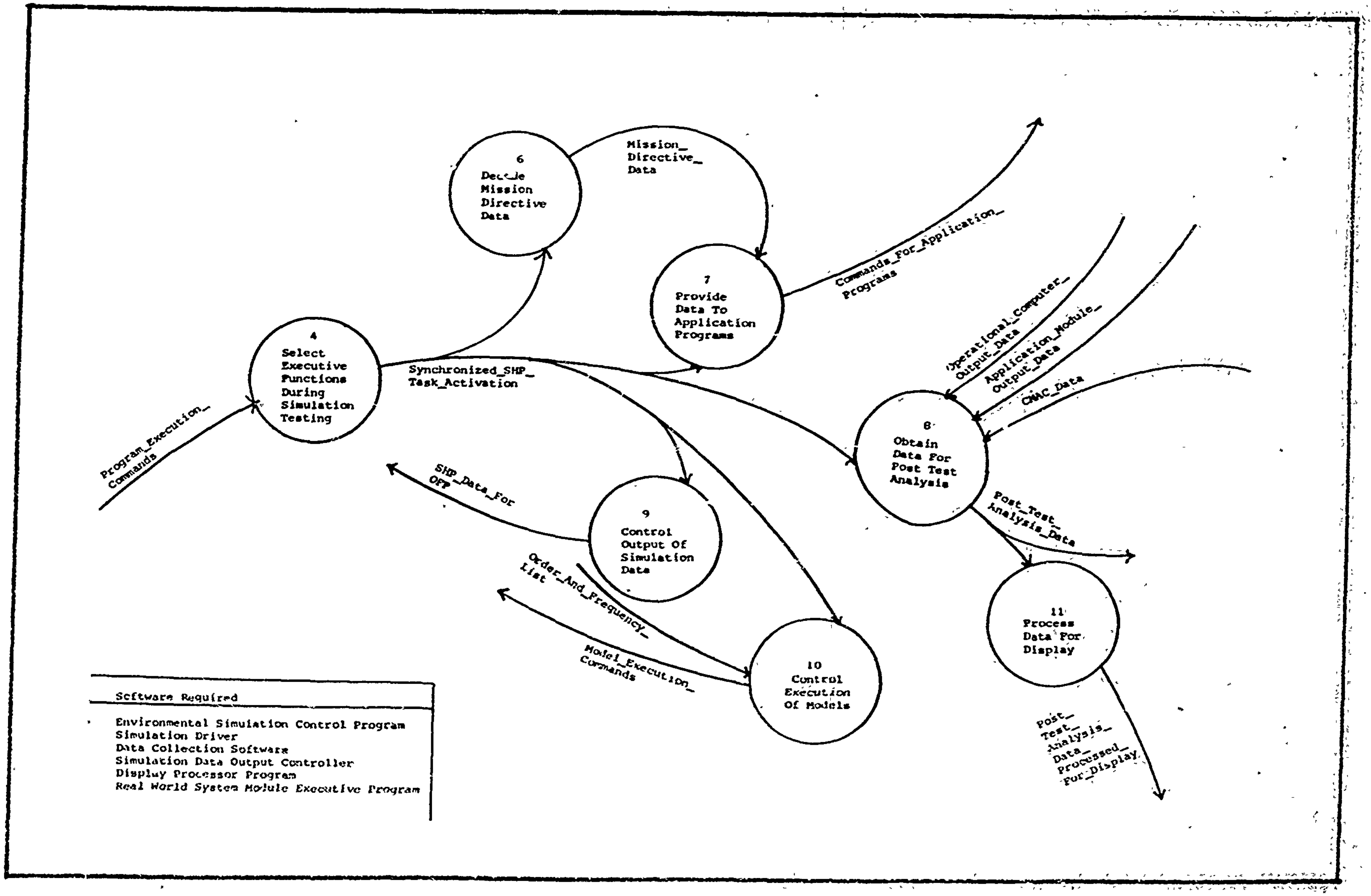


Figure 49. Environmental Simulation Control Program Functions

Figure 46 are for the multi-functional option. The operating system itself is used to support the actual real time processing accomplished on the SHP. Peripheral terminal services are also provided for system initialization and displaying outputs. Other support software tools are used to support interactive processing, batch processing, and files management. Vendor-supplied software tools such as assemblers, compilers, editors, and debuggers are also required for the program generation function.

The next major type of real time support software resident of the SHP is the SHP Executive and Control software. Functions performed by this software are shown in Figure 47. The Simulation Master Ececutive controls the program execution sequence by receiving commands from the SHP operating system and acting upon them by passing control to one of the other software functions. Test Definition software interacts with the operator to obtain data necessary to control the simulation test run. The operator inputs may be manual inputs or the identification of files containing the required data or a combination of both. The Test Preparation Program performs the actions required to initiate a simulation test run. These actions are shown in Figure 48.

Executive functions to be performed during the simulation run are accomplished by the Environmental Simulation Control Program (Figure 49). The Simulation Driver decodes mission directive data and provides data to the Application Programs. Operational computer program output data, application program output data, and CMAC data for post test analysis is obtained

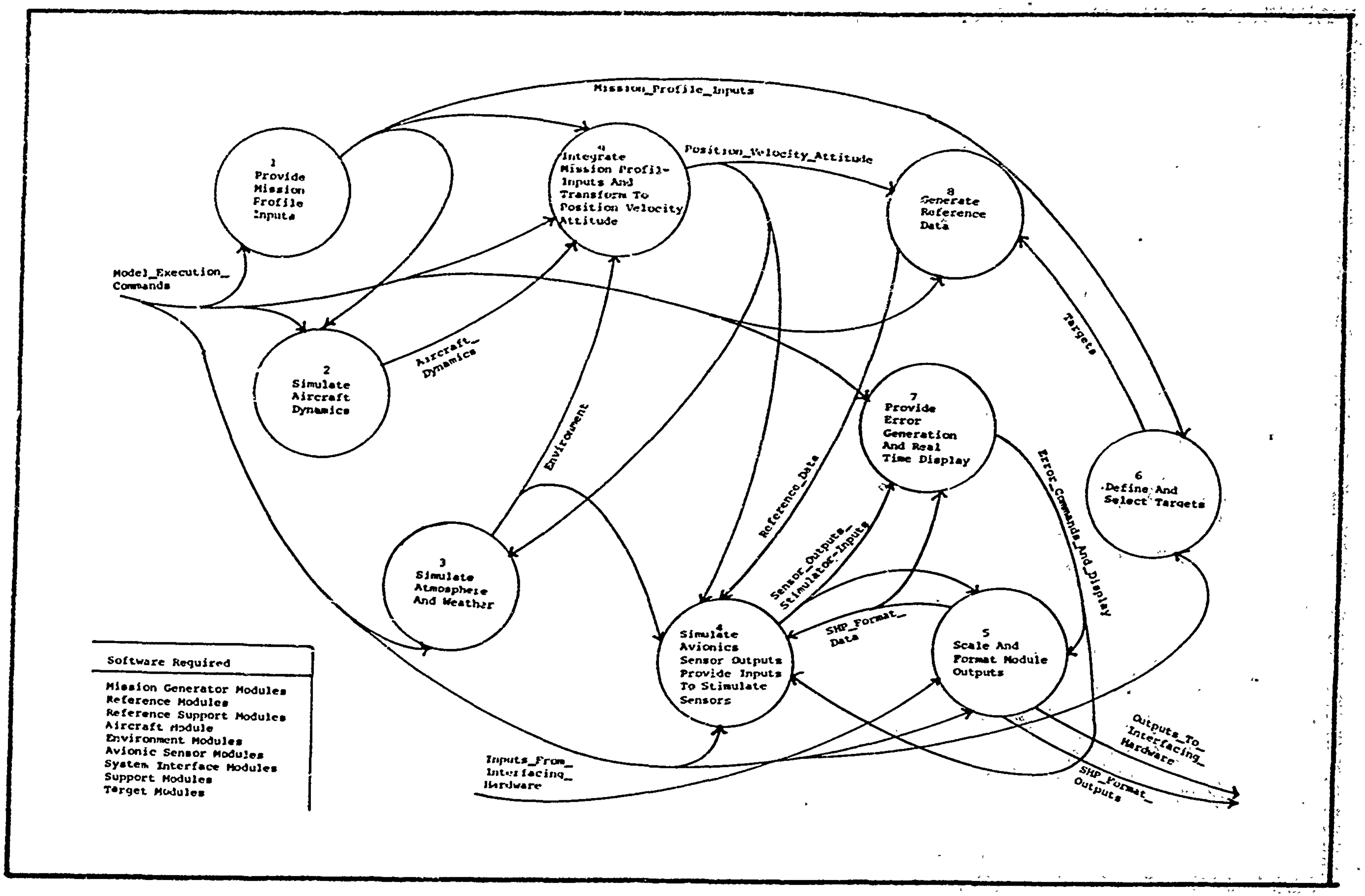
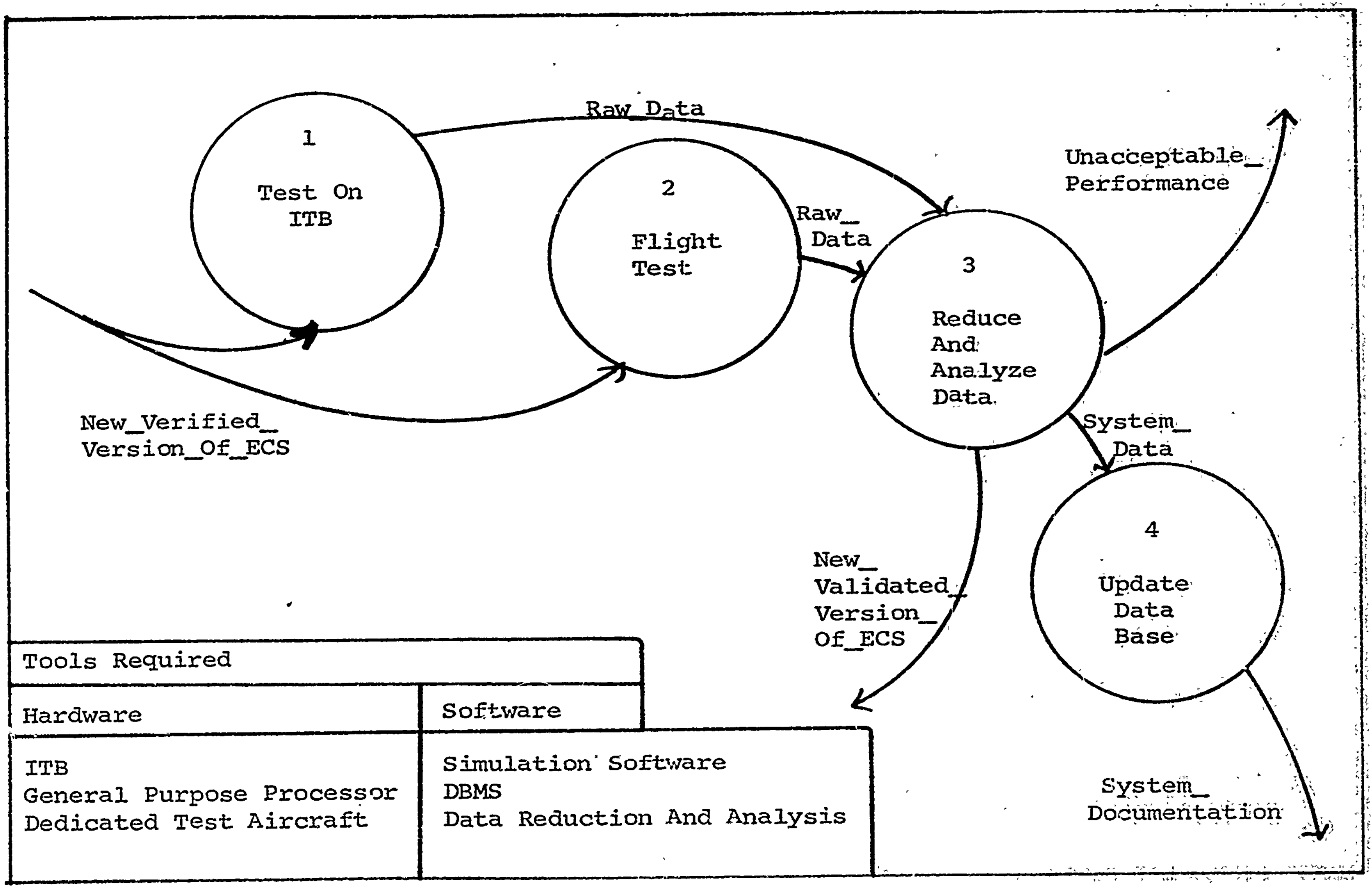

1 Provide Mission Profile Inputs
2 Simulate Aircraft Dynamics
3 Simulate Atmosphere And Weather
4 Integrate Mission Profile Inputs And Transform To Position Velocity Attitude
4 Simulate Avionics Sensor Outputs Provide Inputs To Stimulate Sensors
5 Scale And Format Module Outputs
6 Define And Select Targets
7 Provide Error Generation And Real Time Display
8 Generate Reference Data
Model_Execution_Commands
Mission_Profile_Inputs
Position_Velocity_Attitude
Aircraft_Dynamics
Environment
Reference_Data
Targets
Error_Commands_And_Display
Sensor_Outputs_Stimulator_Inputs
SHP_Format_Data
Inputs_From_Interfacing_Hardware
Outputs_To_Interfacing_Hardware
SHP_Format_Outputs
Software Required
Mission Generator Modules
Reference Modules
Reference Support Modules
Aircraft Module
Environment Modules
Avionic Sensor Modules
System Interface Modules
Support Modules
Target Modules

Figure 50. SHP Application Modules Functional Requirements

by the Data Collection software. This data is then accessed by the Display Processor Software which processes it for display on alphanumeric and/or graphic displays. The Simulation Data Output Controller provides data generated in the SHP to the OFP in the avionics computer in the allocated time window so the inputs can be read into the OFP without delay. Execution of the environment and aircraft subsystem models is controlled by the Real World System Module Executive software, which is list driven. The list defines the order and frequency of the application model's execution. The Application Module functional requirements are shown in Figure 50.

Mission profile inputs are provided to the Aircraft Module, Reference Modules, and the Target Module by the Mission Generator Modules. The Reference Modules accept inputs from the Mission Generators and the Aircraft Module and integrates them transforming them into position, velocity, and aircraft attitude data. The Reference Support Modules accept data from the Reference Modules and generate reference data for comparison with outputs generated by the OFP. The Aircraft Module simulates the particular aircraft's dynamic qualities and provides this data to the Reference Modules. Models simulating atmospheric conditions and weather make up the Environment Modules which provide environment data to the Reference Modules and Avionic Sensor Modules. The Avionic Sensor Modules simulate avionic sensor outputs for those sensors not actually present in the ITB and provide inputs to stimulate those sensors which are present. This data



Tools Required	
Hardware	Software
ITB General Purpose Processor Dedicated Test Aircraft	Simulation Software DBMS Data Reduction And Analysis

Figure 51. System Acceptance

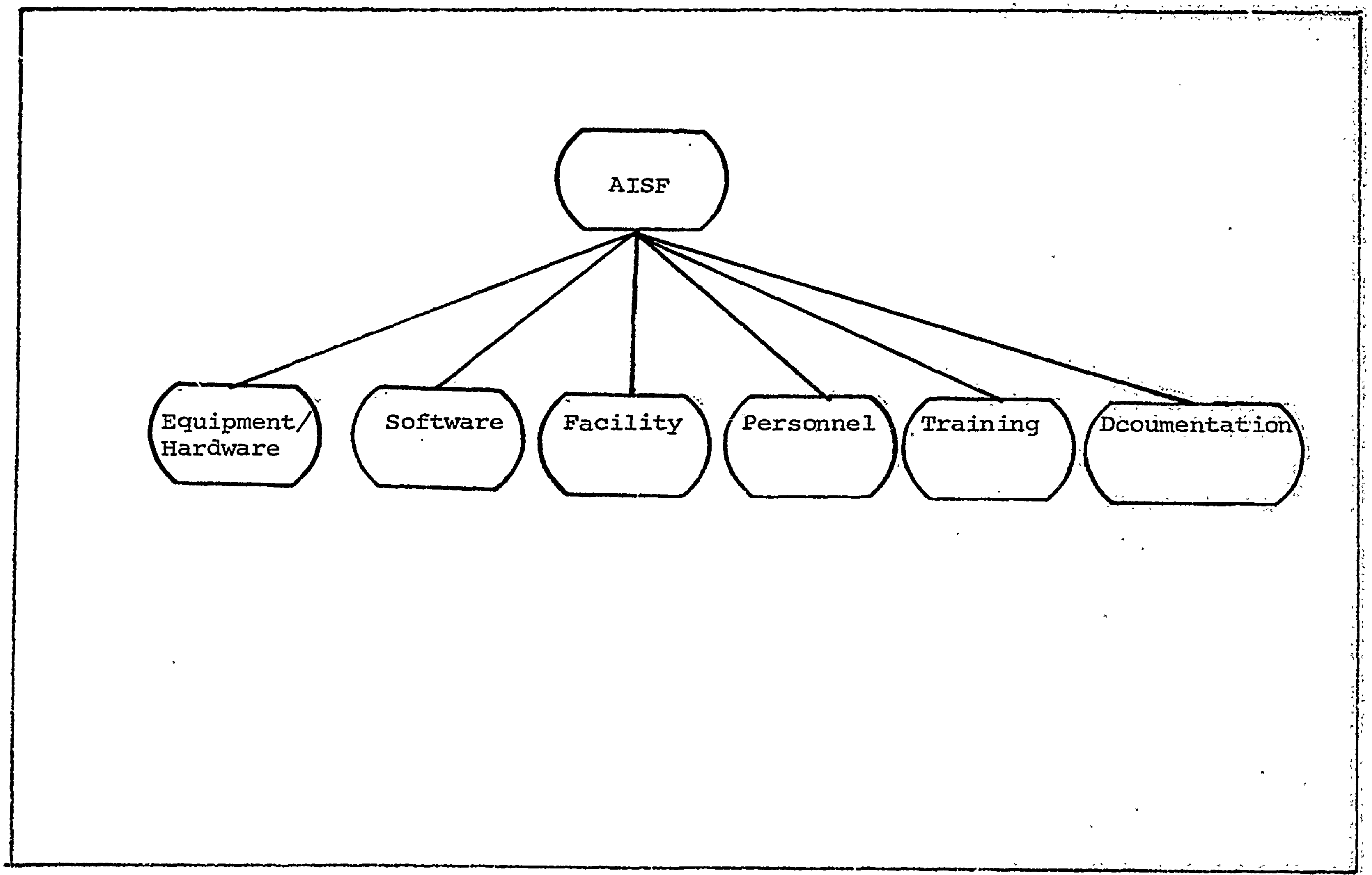


Figure 52. AISF Generic Resources Categories

is provided to the Support and System Interface Modules. The System Interface Modules scale and format outputs from the Support and Avionics Sensor Modules which must be prepared for inputting into the interfacing hardware such as the CMAC. These modules also read inputs from the interfacing hardware and convert them to SMP format. The functions of error generation and real time display are handled by the Support Modules, and the target definition and selection functions are accomplished by Target Modules.

The last functions to be accomplished during the Testing and Verification Phase compose the System Acceptance process. This process is illustrated in Figure 51. The functions of validating the new ECS, both on the ITB and in flight tests, are carried on simultaneously. The raw data produced by these functions are integrated and reduced and analyzed through the techniques described in Figure 31. Once the results of the validation functions have been reduced and analyzed and it has been determined that the new ECS is performing according to specifications determined in the Analysis Phase, the last function, Updating the System Data Base, must be performed. When this function is complete and the new system has been completely documented, the new ECS is ready for use by the operational Air Force.

AISF Generic Resource List

Presented in this section is a hierarchical generic resource list (Ref 5:54) required to implement the functional requirements previously determined. The first, and most high level diagram, Figure 52, illustrates the six basic categories

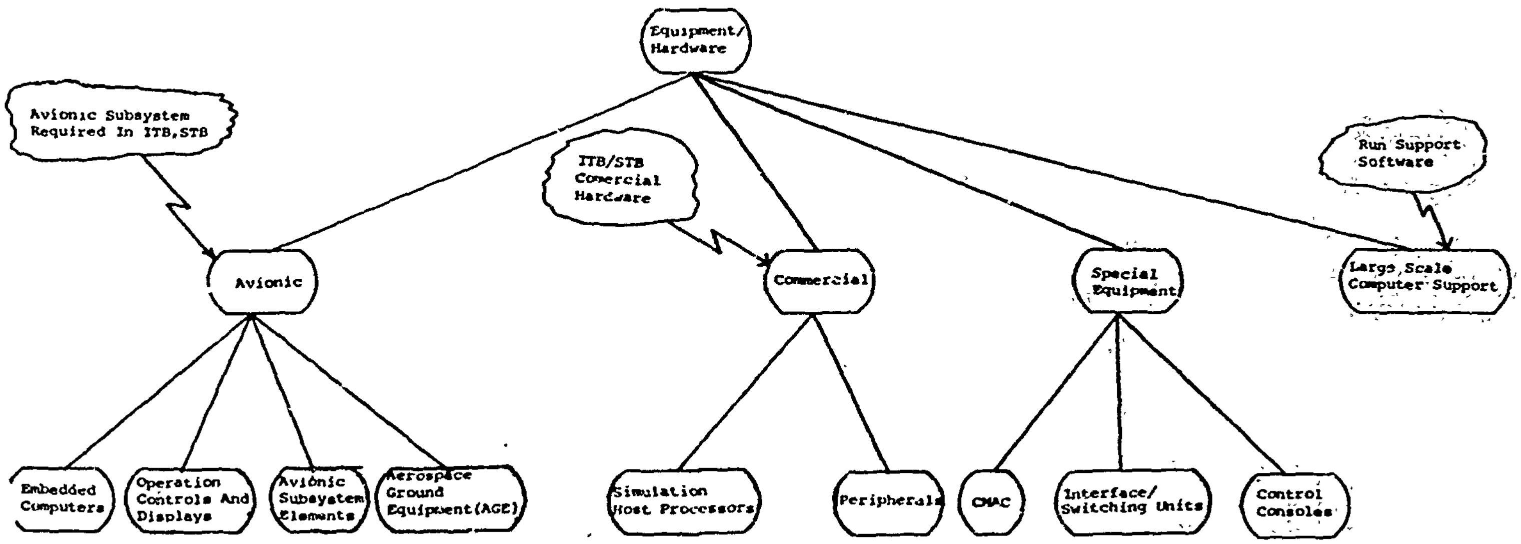


Figure 53. Equipment/Hardware Resources

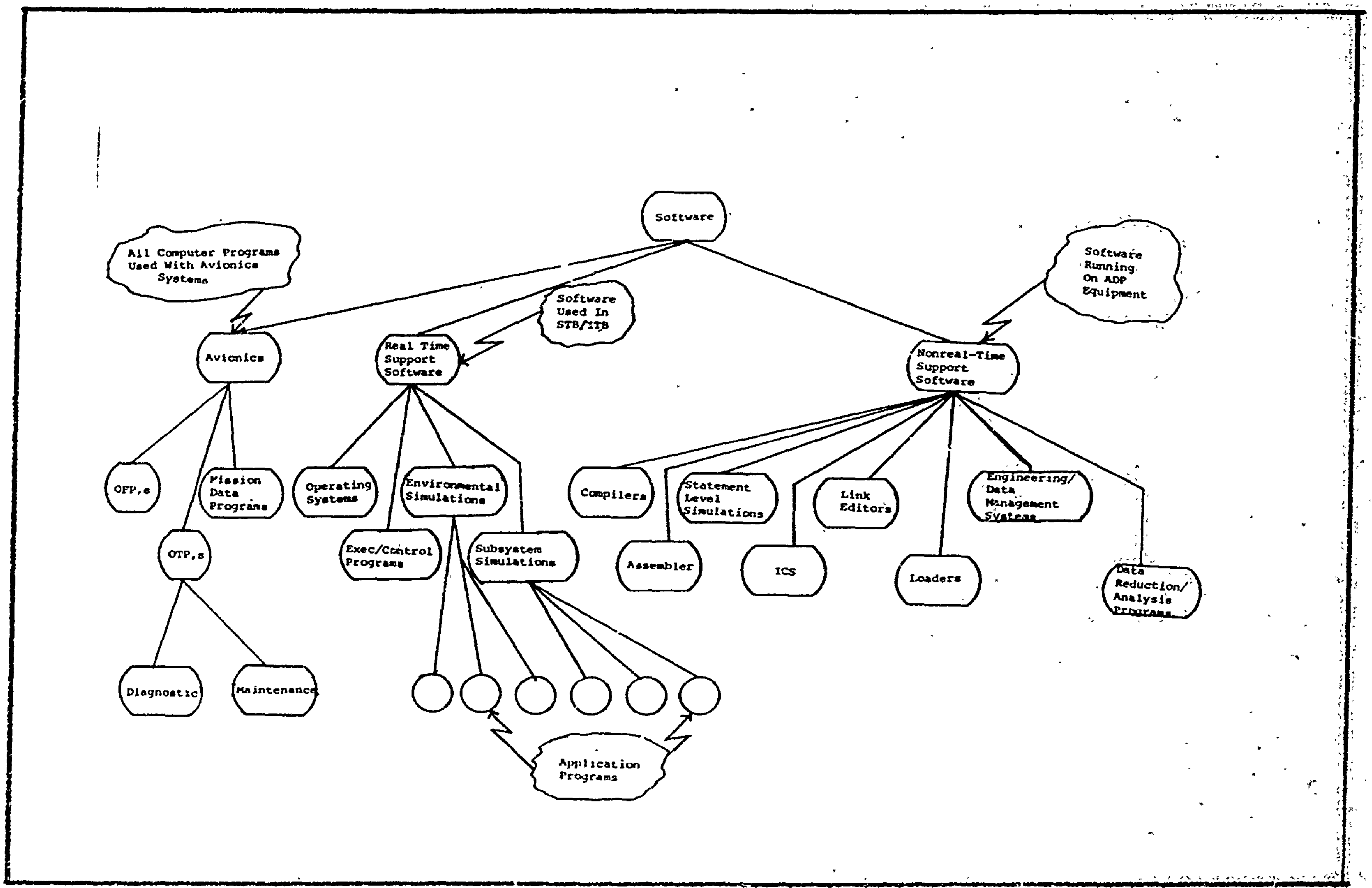

Software
All Computer Programs Used With Avionics Systems
Software Used In STB/ITB
Software Running On ADP Equipment
Avionics
Real Time Support Software
Nonreal-Time Support Software
OFP,s
Mission Data Programs
OTP,s
Diagnostic
Maintenance
Operating Systems
Environmental Simulations
Exec/Control Programs
Subsystem Simulations
Application Programs
Compilers
Assembler
Statement Level Simulations
ICS
Link Editors
Loaders
Engineering/ Data Management Systems
Data Reduction/ Analysis Programs

Figure 54. AISF Software Resources

of resources required: Equipment/Hardware, Software, Facility, Personnel, Training, and Documentation.

The first category, Equipment/Hardware is detailed in Figure 53. The hardware can be subdivided into four major categories. The Avionic Hardware includes the avionics computer(s) and subsystems. This equipment forms a part of the AISF STB and ITB. Elements of avionic hardware used in the test beds include the actual computers, controls and displays, avionic subsystem elements such as sensors, and Aerospace Ground Equipment (AGE). Commercial computer equipment is also employed in the test beds and includes Simulation Host Processors (SHPs) and their peripherals. Special equipment required by the AISF includes CMAC units for test bed diagnostics, interface and switching units for interfacing SHPs and avionics computers, and control consoles for operating test beds. The last category of hardware required for an AISF includes large scale computer support hardware required for hosting support software such as assemblers, compilers, and DBMSs.

Software resources required for AISF implementation consists of three categories shown in Figure 54. The first category, Avionics Software is made up of all computer programs used by the avionics computer systems. These programs include the OFP, Operational Test Programs (OTPs) used for diagnostic and maintenance functions, and various Mission Data Programs. The second category of software resources consists of the Real Time Support Software employed in the test beds. Real Time Support Software consists of Operating Systems, Executive anc Control programs, Environmental Simulations, and Subsystem

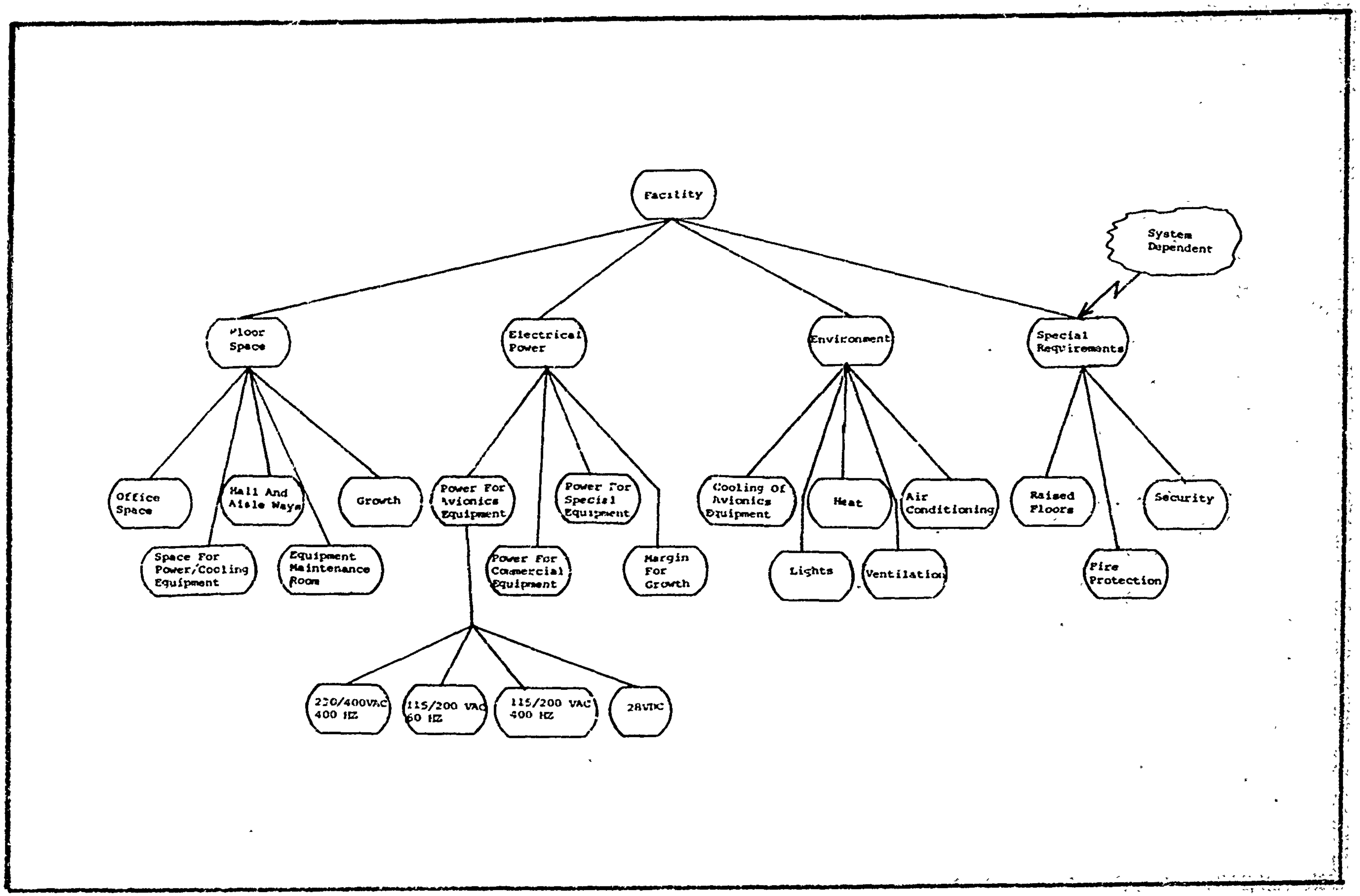


Figure 55. Facility Resources

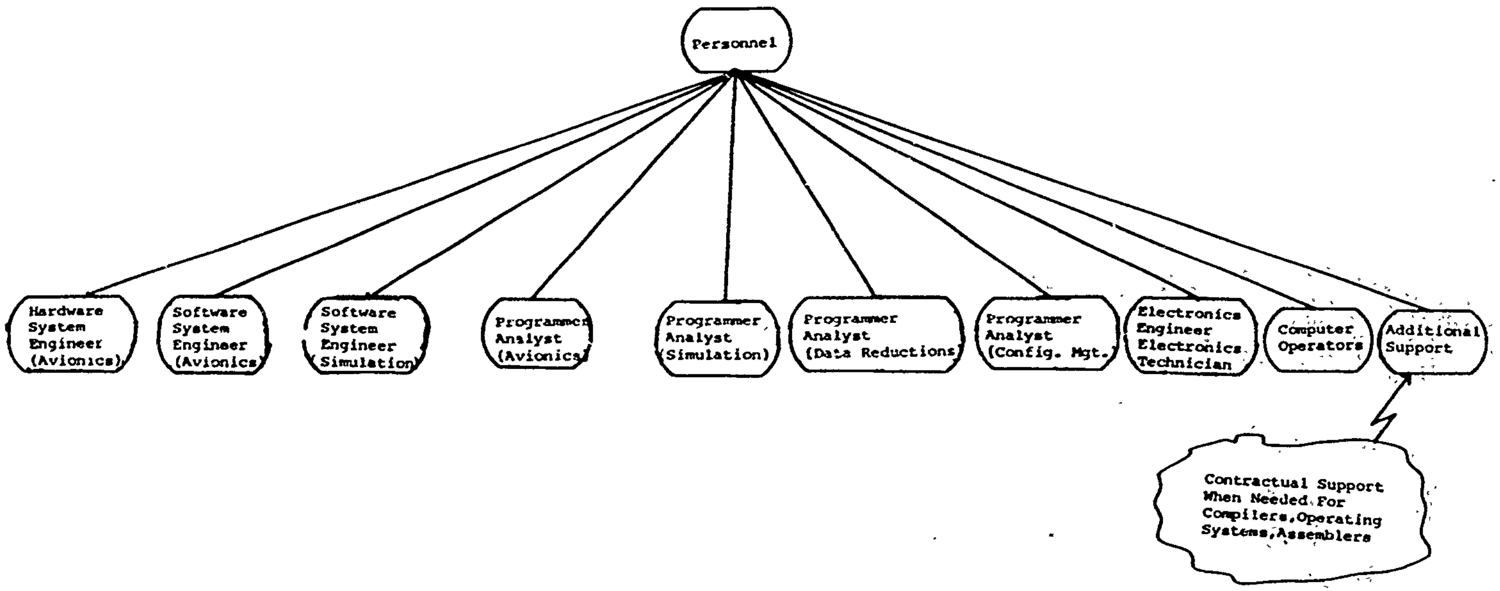

Personnel
Hardware System Engineer (Avionics)
Software System Engineer (Avionics)
Software System Engineer Simulation)
Programmer Analyst (Avionics)
Programmer Analyst (Simulation)
Programmer Analyst (Data Reductions)
Programmer Analyst (Config. Mgt.
Electronics Engineer Electronics Technician
Computer Operators
Additional Support
Contractual Support When Needed For Compilers, Operating Systems, Assemblers

Figure 56. Personnel Resources

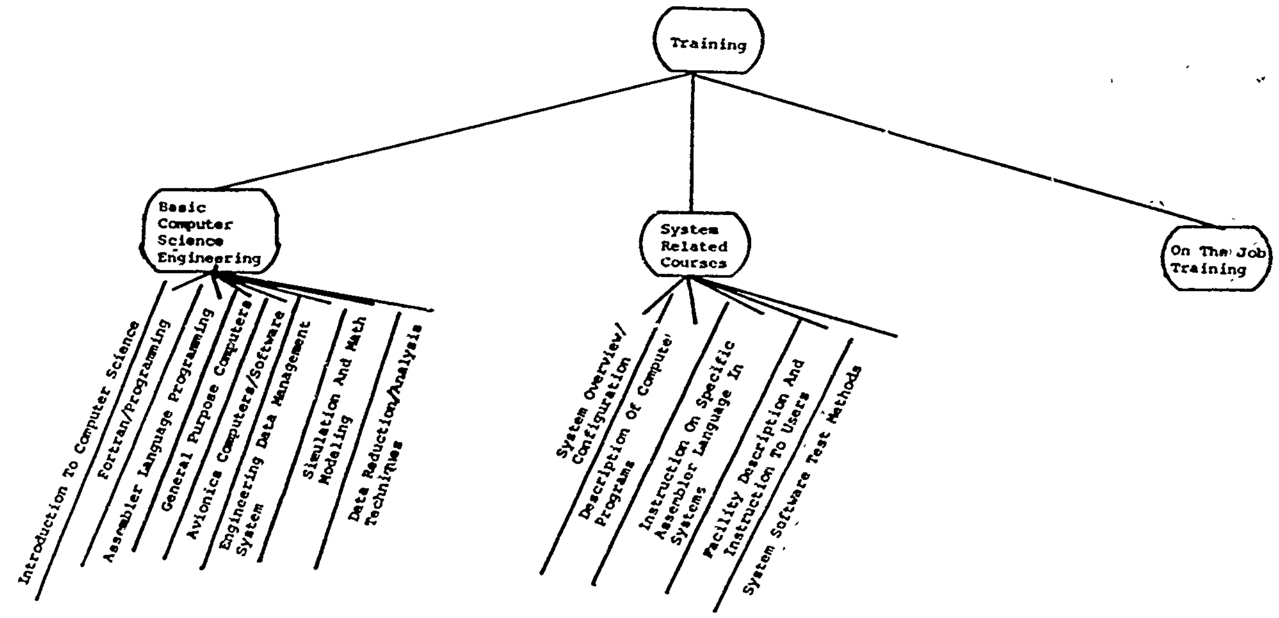

Training
Basic Computer Science Engineering
System Related Courses
On The Job Training
Introduction To Computer Science
Fortran/Programming
Assembler Language Programming
General Purpose Computers
Avionics Computers/Software
Engineering Data Management System
Simulation And Math Modeling
Data Reduction/Analysis Techniques
System Overview/ Configuration
Description Of Computer Programs
Instruction On Specific Assembler Language In Systems
Facility Description And Instruction To Users
System Software Test Methods

Figure 57. Personnel Training Resources

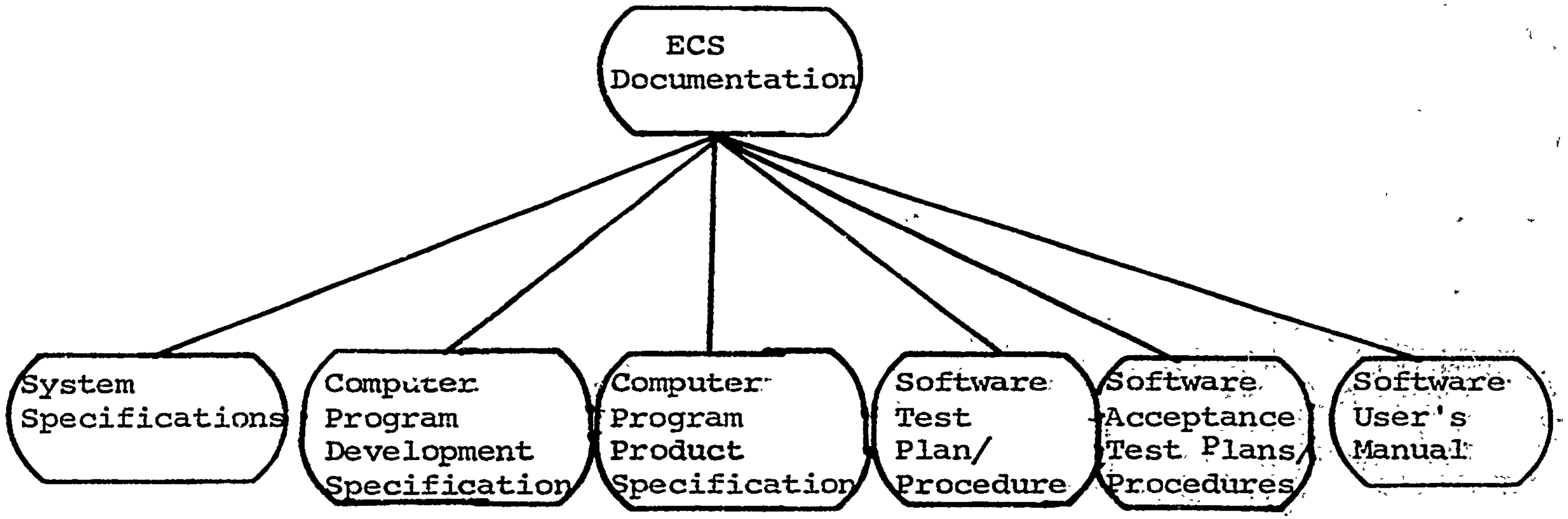

ECS Documentation
System Specifications
Computer Program Development Specification
Computer Program Product Specification
Software Test Plan/ Procedure
Software Acceptance Test Plans/ Procedures
Software User's Manual

Figure 58. Documentation Resources

Simulations. These simulations are made up of modules which each simulate a portion of the environment or a subsystem. The last category of software resources consists of Non-Real Time Support Software. These are programs run on the General Purpose Processor and include compilers, assemblers, various simulations, link editors, loaders, DBMSs, and Data Reduction and Analysis programs.

Facility resources shown in Figure 55 consist of the actual structrues which house the AISF equipment. Facility considerations include floor space, electrical power requirements, environmental considerations, and special requirements such as security and fire protection devices.

Figure 56 illustrates the various personnel and skills required for AISF implementation. The majority of skills are in the electrical engineering and computer science disciplines. Additional support includes contractual personnel to support vendor supplied equipment and software. The personnel will require special training of the type shown in Figure 57. This figure depicts some of the recommended courses to prepare personnel for working in an AISF environment.

Figure 58 illustrates the ECS documentation required for the effective operation of an AISF. Some of this documentation is acquired from outside sources such as the ECS system specification, and some is AISF generated such as the software test plan procedure.

Summary

In this chapter the process discussed in Chapter II was applied to render functional requirements for an Avionics Integration Support Facility.

Evolution of the functional requirements involved tradeoffs at all levels. The initial problem considered was that of providing the best, most cost-effective, support for the Air Force's ECS software. The goal was to allow the Air Force to maintain maximum control over OFP changes while holding contractual support to a minimum. The facility must incorporate both system engineering and computer software expertise and must provide for verification and validation of OFP changes. The fact that OFPs are not presently transferrable between ECSs played a restrictive role in the functional requirements development. The intent was to develop functional requirements for a standard AISF. Though the functional requirements presented here are universal among AISFs, the facility implementation is still system-dependent. Present trends towards avionics hardware and software standardization were incorporated in the development of the functional requirements to the extent possible and will, no doubt, play an important role in the development of a truly standard, ECS-independent, AISF in the future. Chapter VII of this investigation is dedicated solely to this subject.

Similar constraints and tradeoffs drove the evolution of each phase. Of particular importance in the Analysis Phase was the need to utilize automated design and analysis tools in analyzing change feasibility and defining change requirements. Similarly, automated tools and a structured approach were crucial to the effective accomplishment of

Design and Mechanization. Proper hardware and associated software tools are required to develop a test plan and test criteria and verify the design. Tradeoffs in determining the SHP to be used for this phase and the Verification and Test Phase were also considered. Whether the SHP is to be strictly dedicated to real time simulation or to both real time simulation and other software development activities became an important factor. If the SHP is strictly dedicated to real time simulation, the General Purpose Processor must accomodate all other data processing tasks. If the SHP's functional requirements allow it to also be utilized for other functions, the role of the General Purpose Processor is impacted and a machine with less capability may be appropriate.

The method of validation used also presented tradeoffs. The ideal, yet most costly method, is flight testing. This activity requires the use of a dedicated, instrumented, flight test aircraft to perform a job which can be partially accomplished on a much less expensive Hot Bench. As the cost of aircraft and fuel rise, and the cost of hardware drops, the Hot Bench approach becomes a more attractive validation option.

Evaluation of tradeoffs such as these produced the functional requirements which were presented in Data Flow Diagram format. They were presented first at a high level of abstraction, then iterated through levels of detail. The similarity between the FSED process for producing software and the OFP change process accomplished at the AISF was discussed. A generic resourcos list for implementing an AISF was also provided. In the next chapter the functional requirements presented here are modeled with SADT to produce the AISF design.

IV. SADT Actigram Model of AISF Design

Introduction

In this chapter the functional requirements developed in Chapter II are transformed into a design for implementation. The design is presented in the form of a model documented with the Structured Analysis and Design Technique (SADT). SADT was chosen as the vehicle for developing the AISF design because it is a coherent, integrated, set of methods and rules constituting a disciplined approach to analysis and design. Analysis of the problem is top-down, modular, hierarchic, and structured. The SADT language is a diagramming technique showing component parts, interrelationships between them, and how they fit into a hierarchic structure. (Ref 7:21) A short tutorial on SADT is presented in Appendix B. SADT is well suited for documenting the design because it not only illustrates the inputs, functions, and outputs of each process, but also allows the controls and mechanisms by which the process is to be implemented, to be shown on the model.

For the Analysis, Design and Mechanization, and Code and Unit Test Phases, only the high level diagrams are given here. The complete model is contained in Appendix C. The model of the portion of the Test and Verification Phase discussed in Chapters V and VI, the Integration Test Bed (ITB), is shown here in detail.

As shown in the Generic Resources List of Chapter III, an AISF includes a combination of hardware, software, and people. The SADT model analyzes the purposes the AISF components

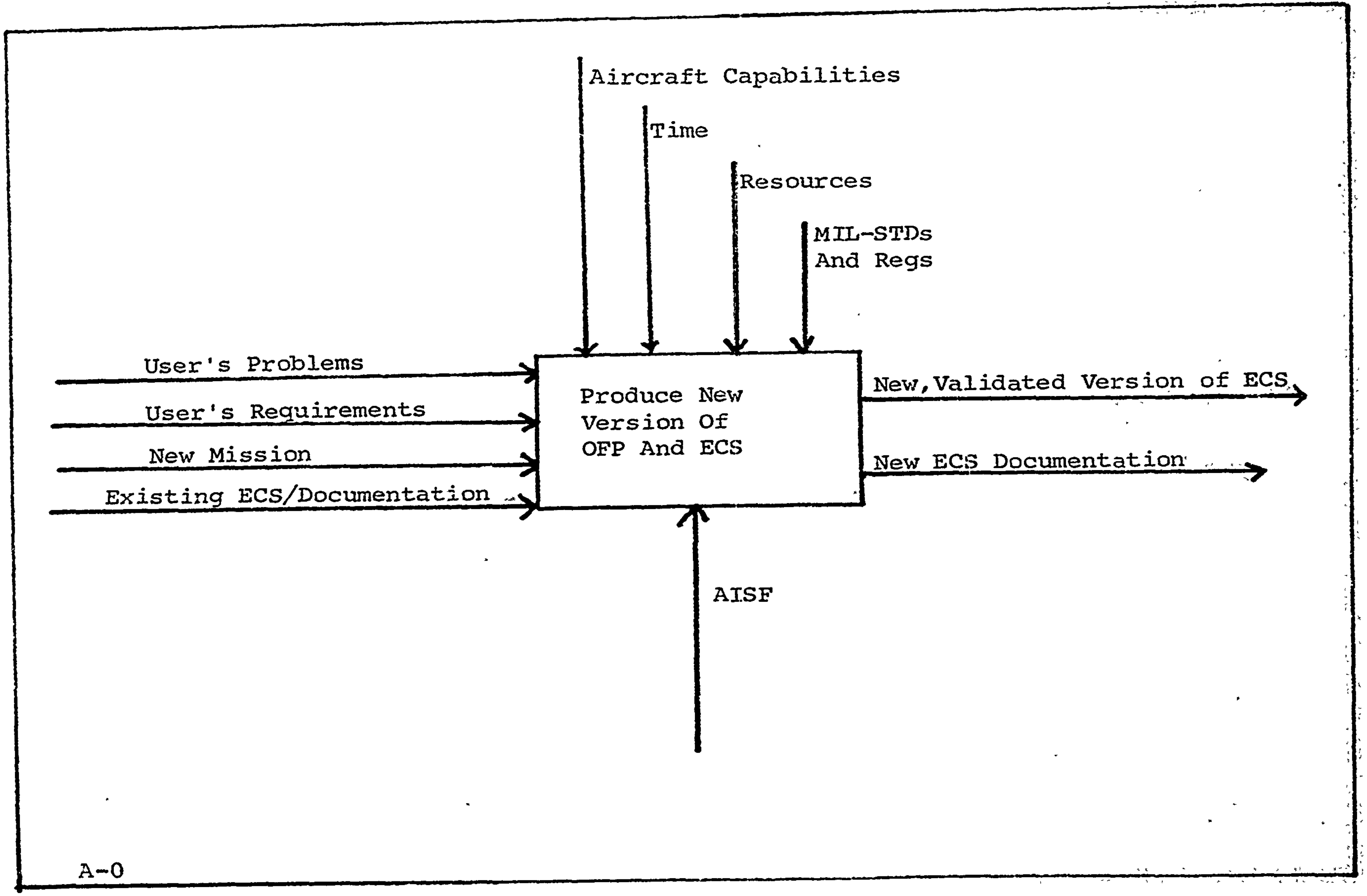


Figure 59. AISF Context

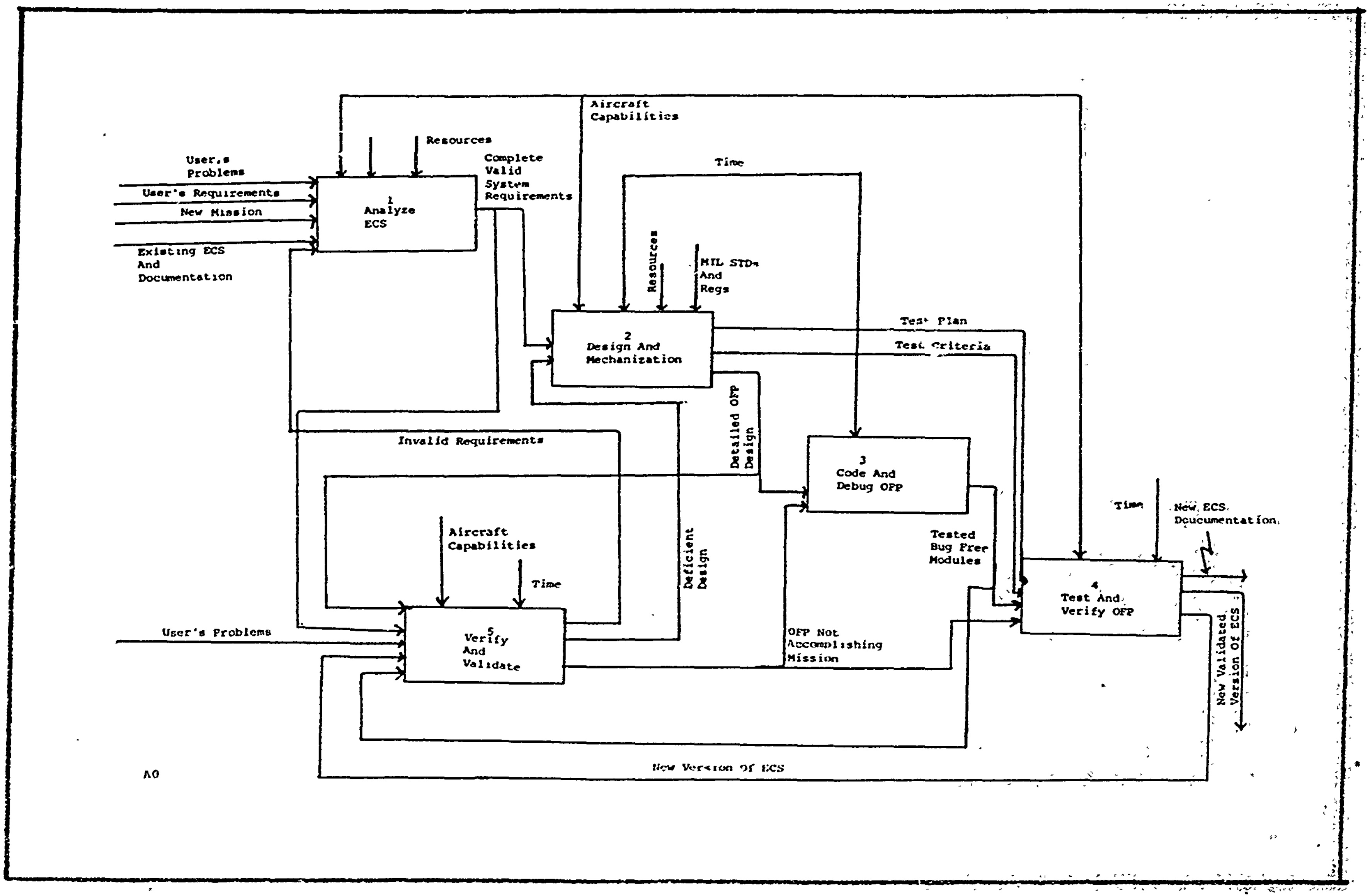


Figure 60. Major Functional Requirements For AISF

serve, the functions they perform, and records the mechanisms by which these purposes and functions are realized.

Context

Figure 59 provides the overall context for the remainder of the model. A hierarchical listing of the model is provided in Table 5.

Table 5

SADT Model Hierarchy

A-0	Context
A0	Major Functional Requirements For AISF
A1	ECS Analysis
A2	Design and Mechanization
A3	Code and Debug OFP
A4	Test and Verify OFP
A41	Integrate and Test ECS
A411	Integrate Software
A412	Test Software
A413	Analyze Program Test Results
A414	Integrate System
A4143	Implement Integration Test Bed
A41431	Provide SHP Services
A41432	Provide For Environment and Avionics Actions
A41433	Implement Interface and Diagnostics
A41434	Implement Human Interface
A415	Test System
A416	Analyze ECS Test Results

The basic function of the AISF is to produce a new version of the OFP and ECS. Inputs to this process are activities which identify the need for the change such as user's problem reports and a new mission for the ECS. The desired outputs of the process are the new, validated, version of the ECS with documentation. Controlling this process are factors such as the target aircraft's capabilities, standards, and regulations.

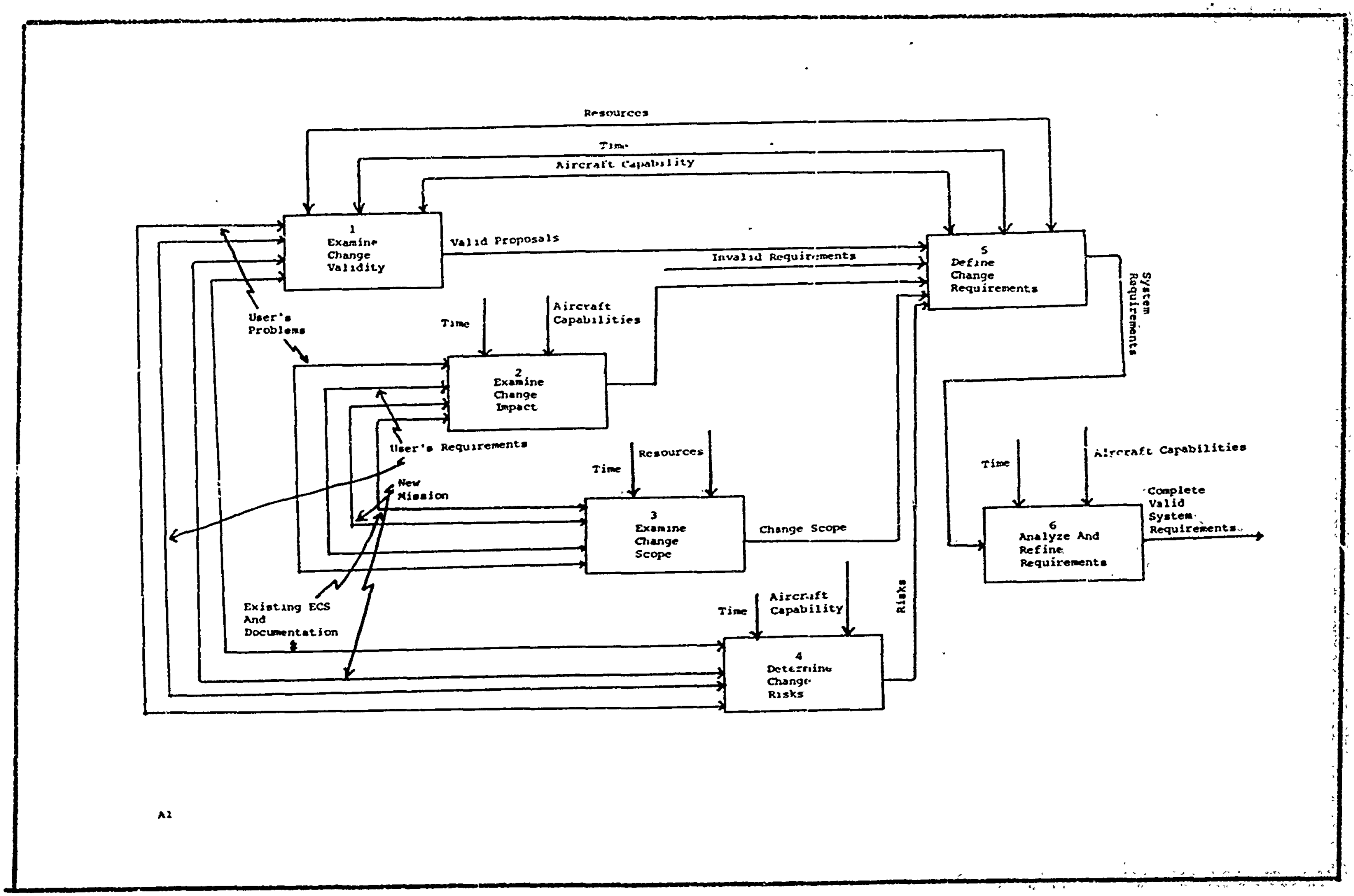


Figure 61. Analyze ECS

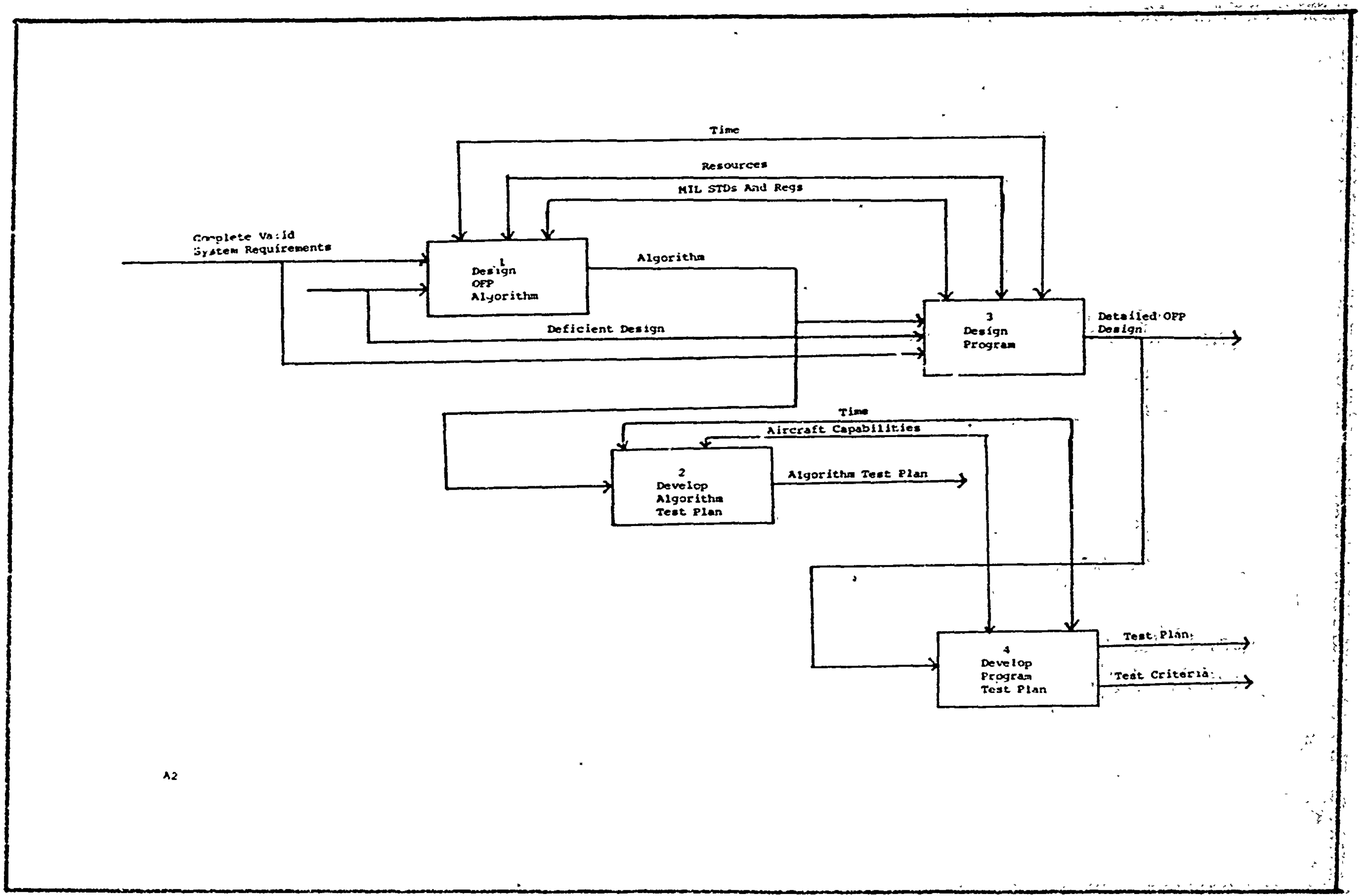


Figure 62. Design And Mechanization

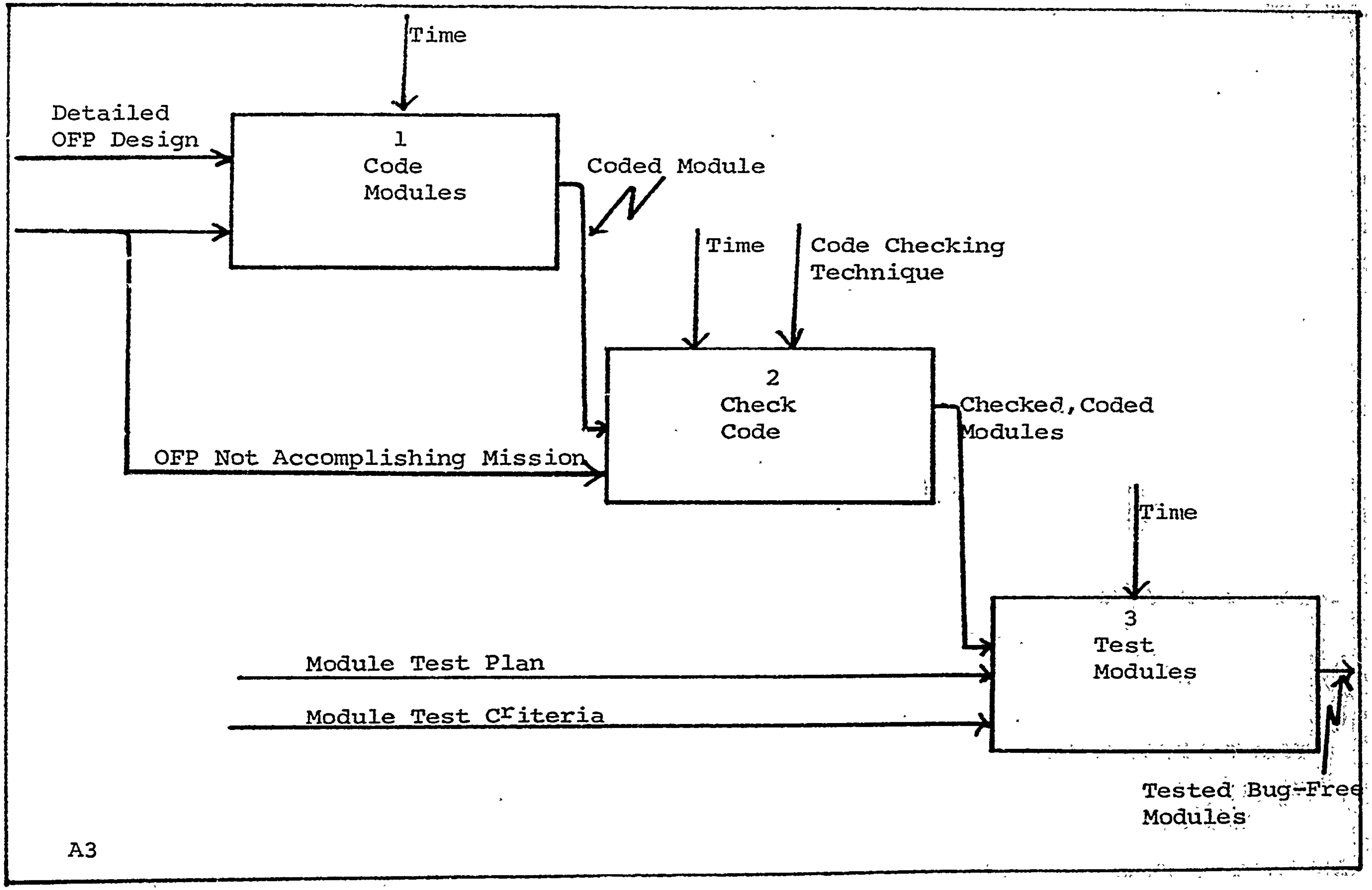


Figure 63. Code And Unit Test

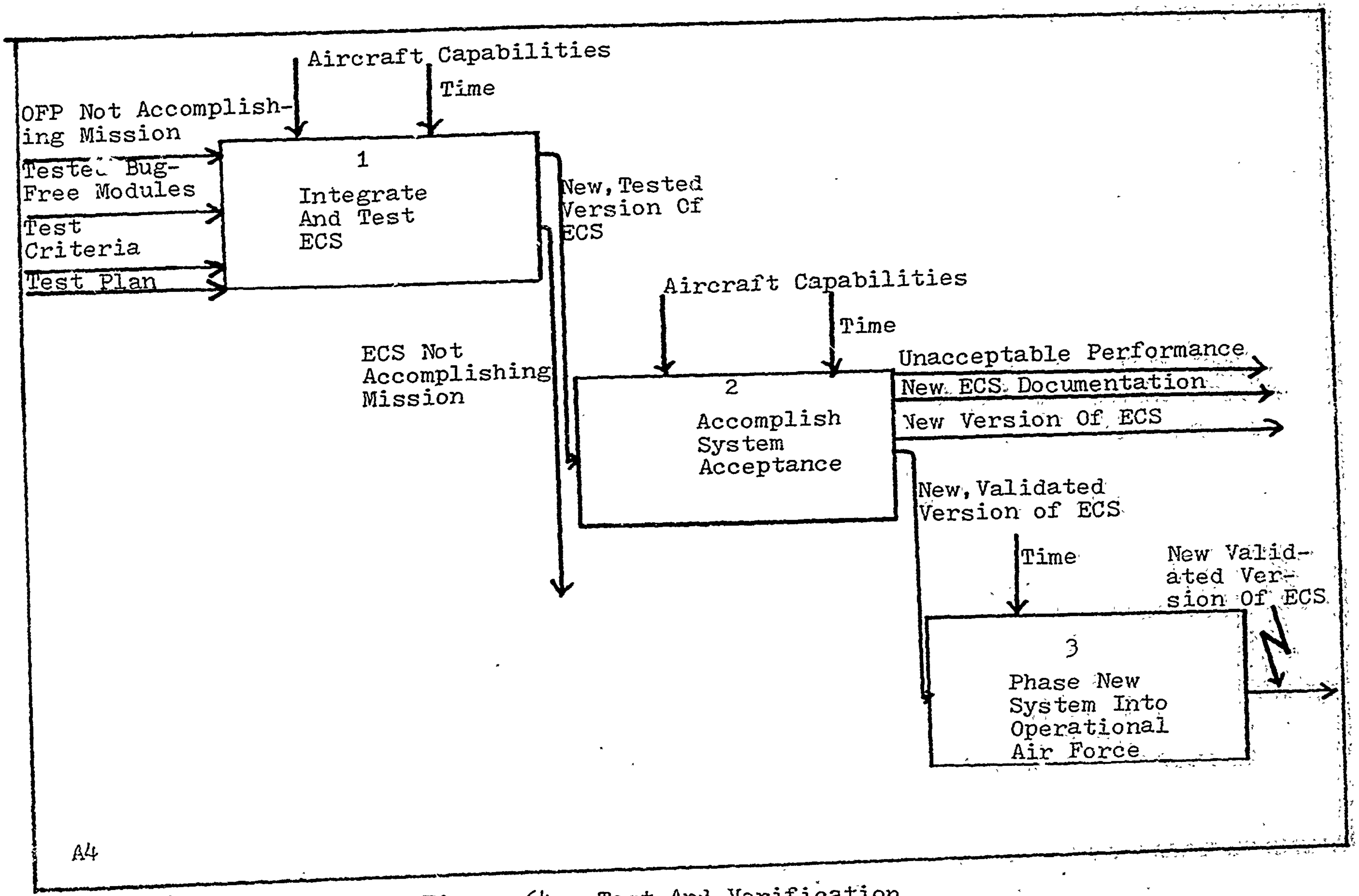

Aircraft Capabilities
Time
OFP Not Accomplishing Mission
Testec Bug-Free Modules
Test Criteria
Test Plan
1
Integrate And Test ECS
New, Tested Version Cf ECS
ECS Not Accomplishing Mission
Aircraft Capabilities
Time
2
Accomplish System Acceptance
Unacceptable Performance
New ECS Documentation
New Version Of ECS
New, Validated Version of ECS
Time
3
Phase New System Into Operational Air Force
New Validated Version Of ECS
A4

Figure 64. Test And Verification

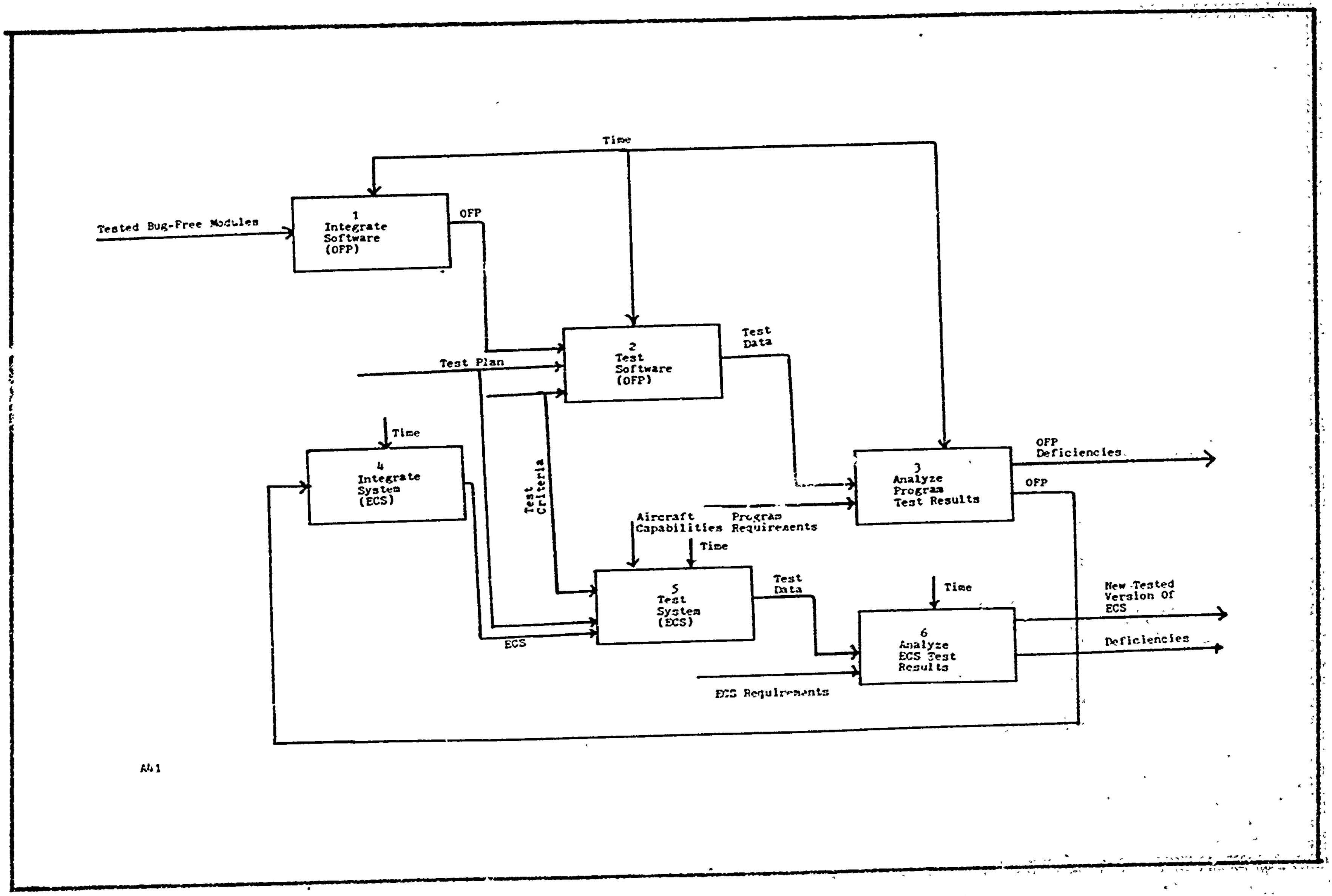


Figure 65. Integrate And Test ECS

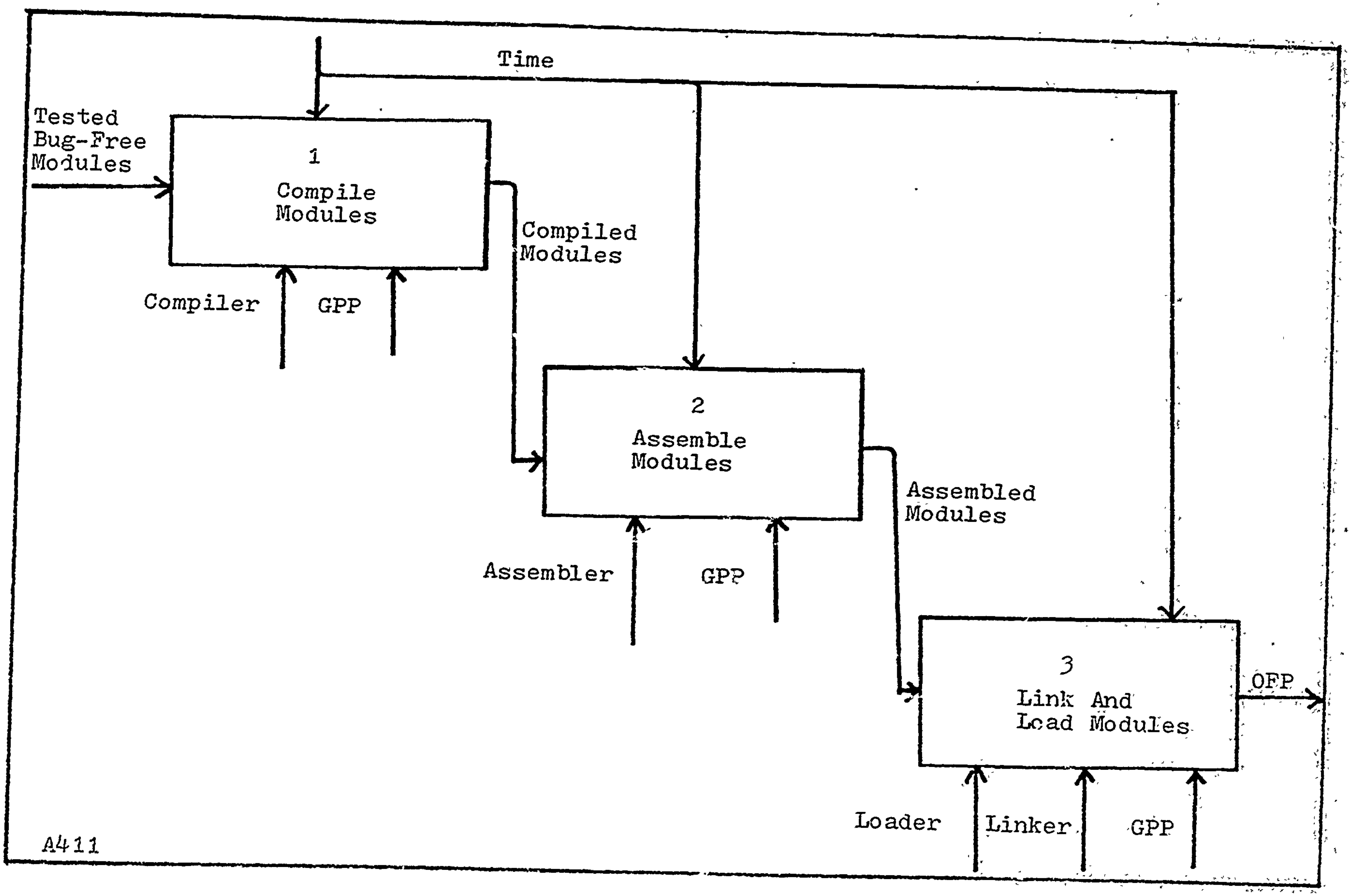


Figure 66. Integrate Software

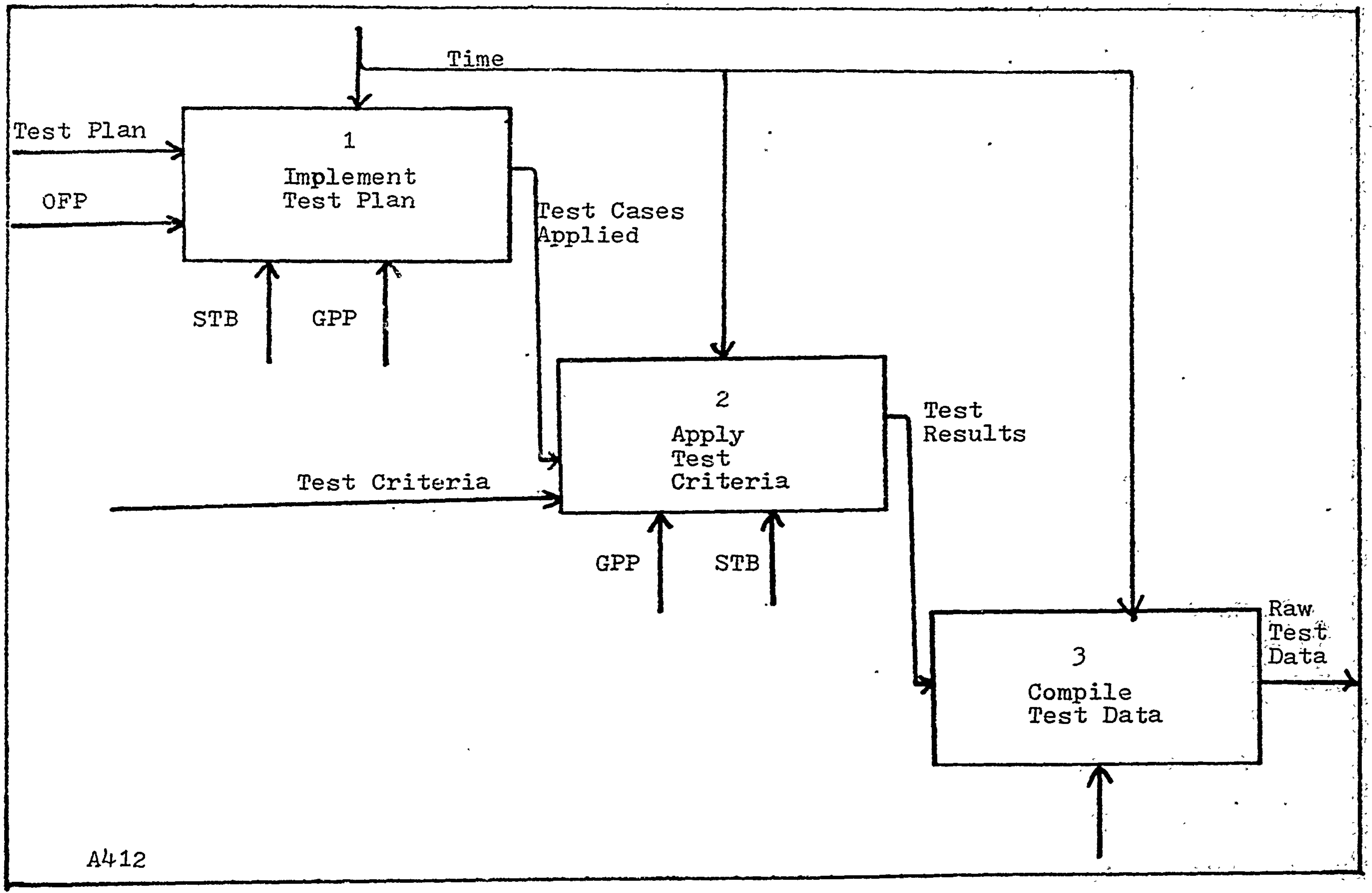


Figure 67. Test Software

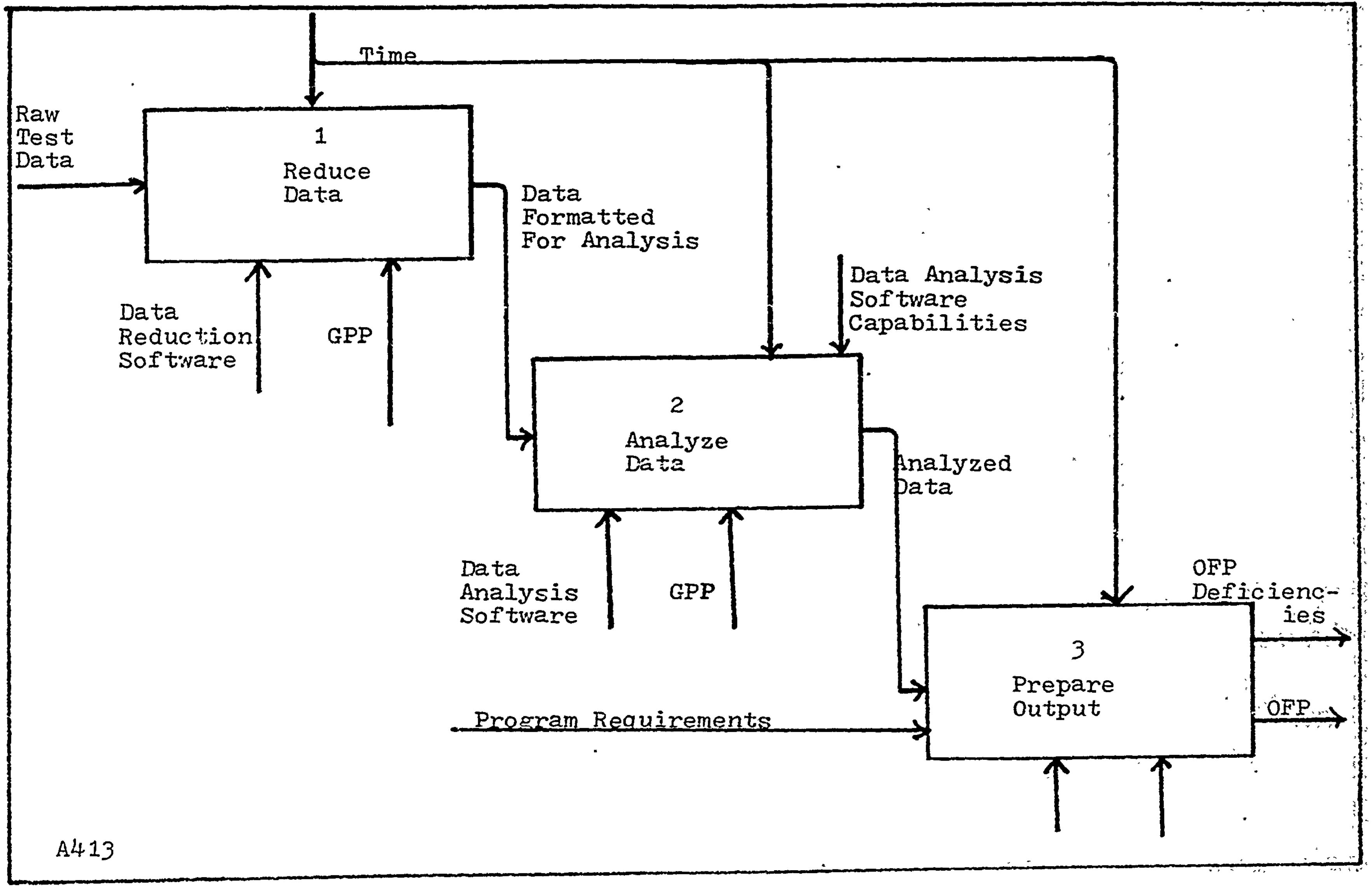


Figure 68. Analyze Program Test Results

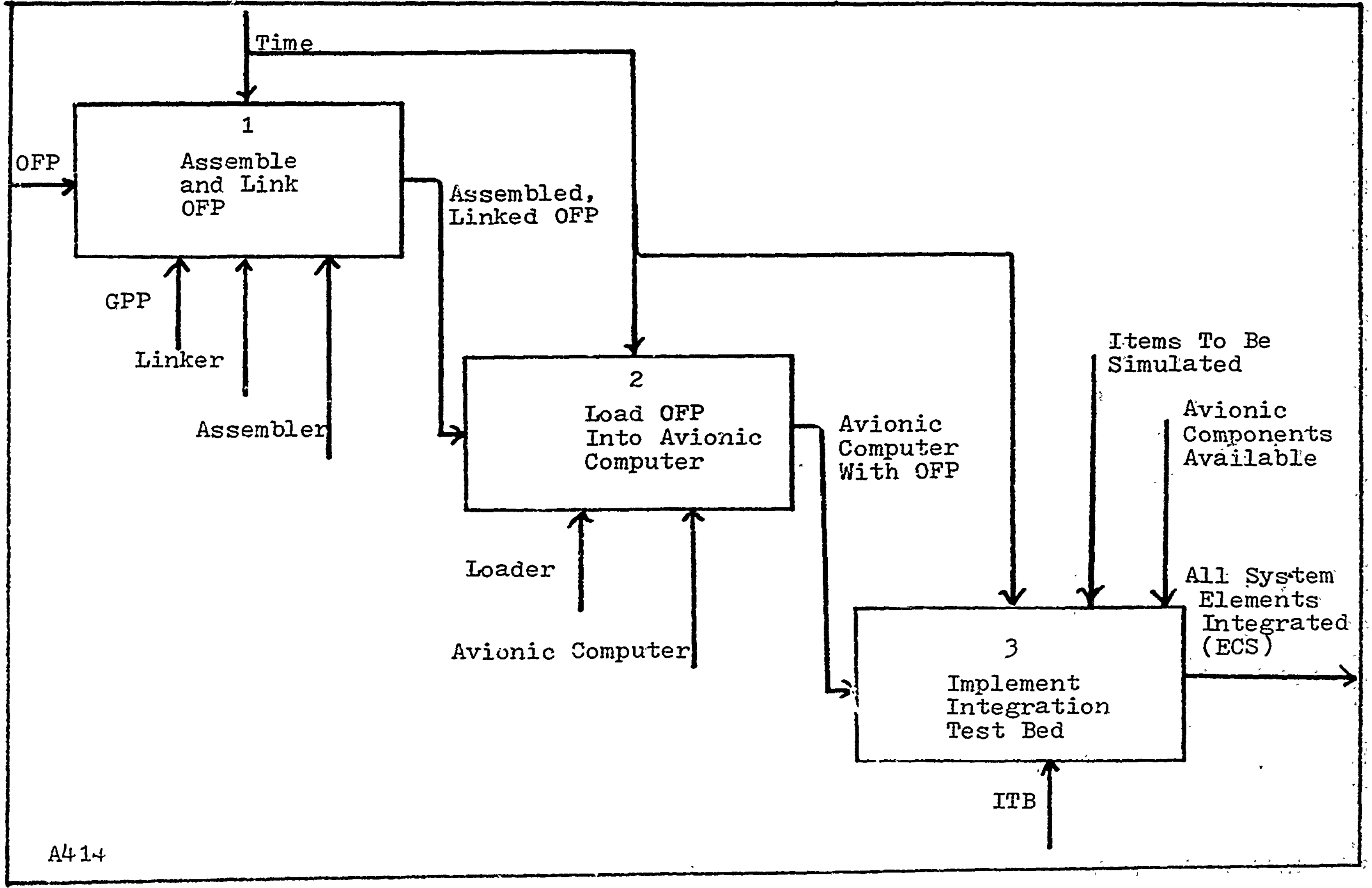


Figure 69. Integrate System

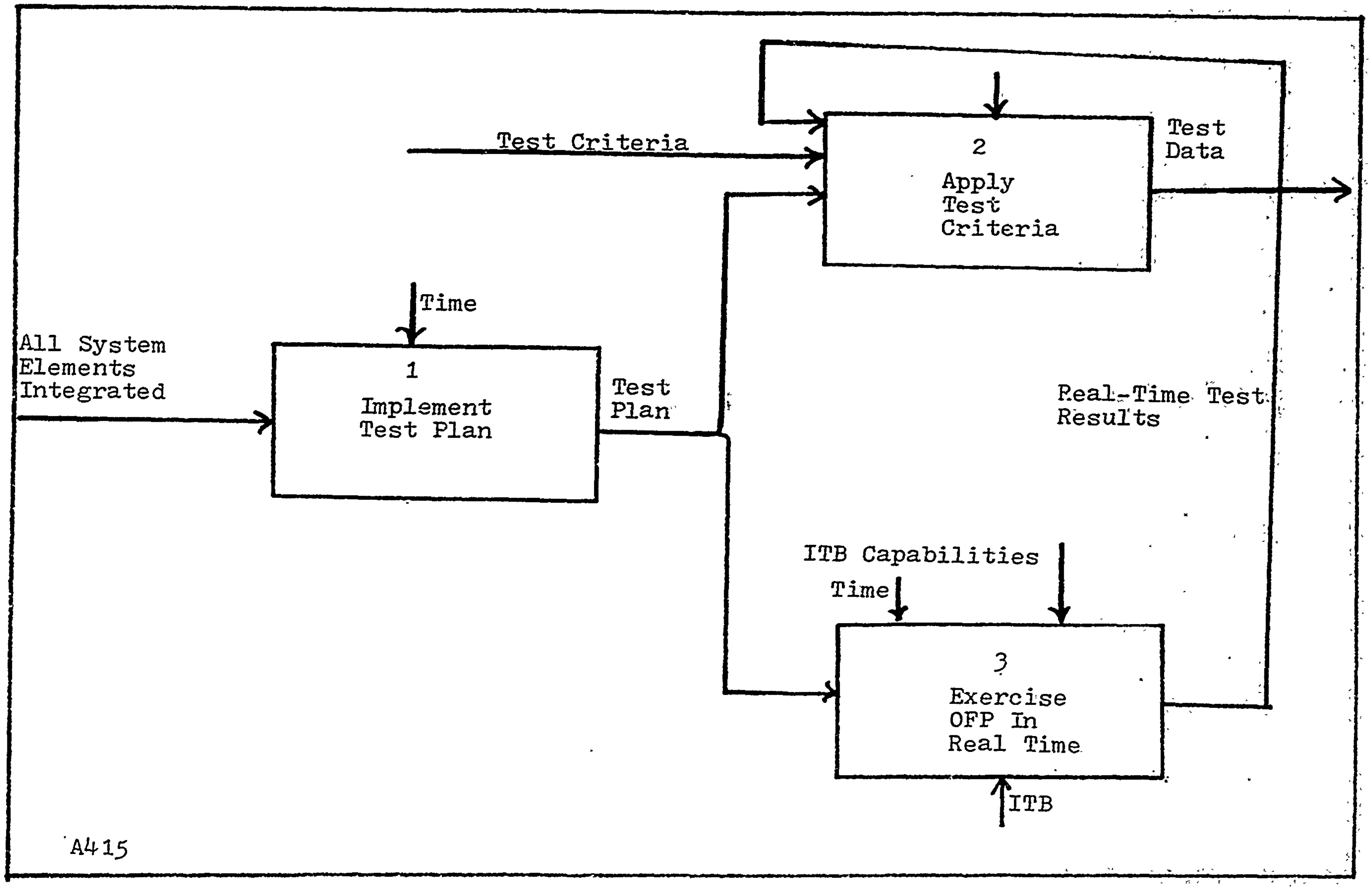


Figure 70. Test System

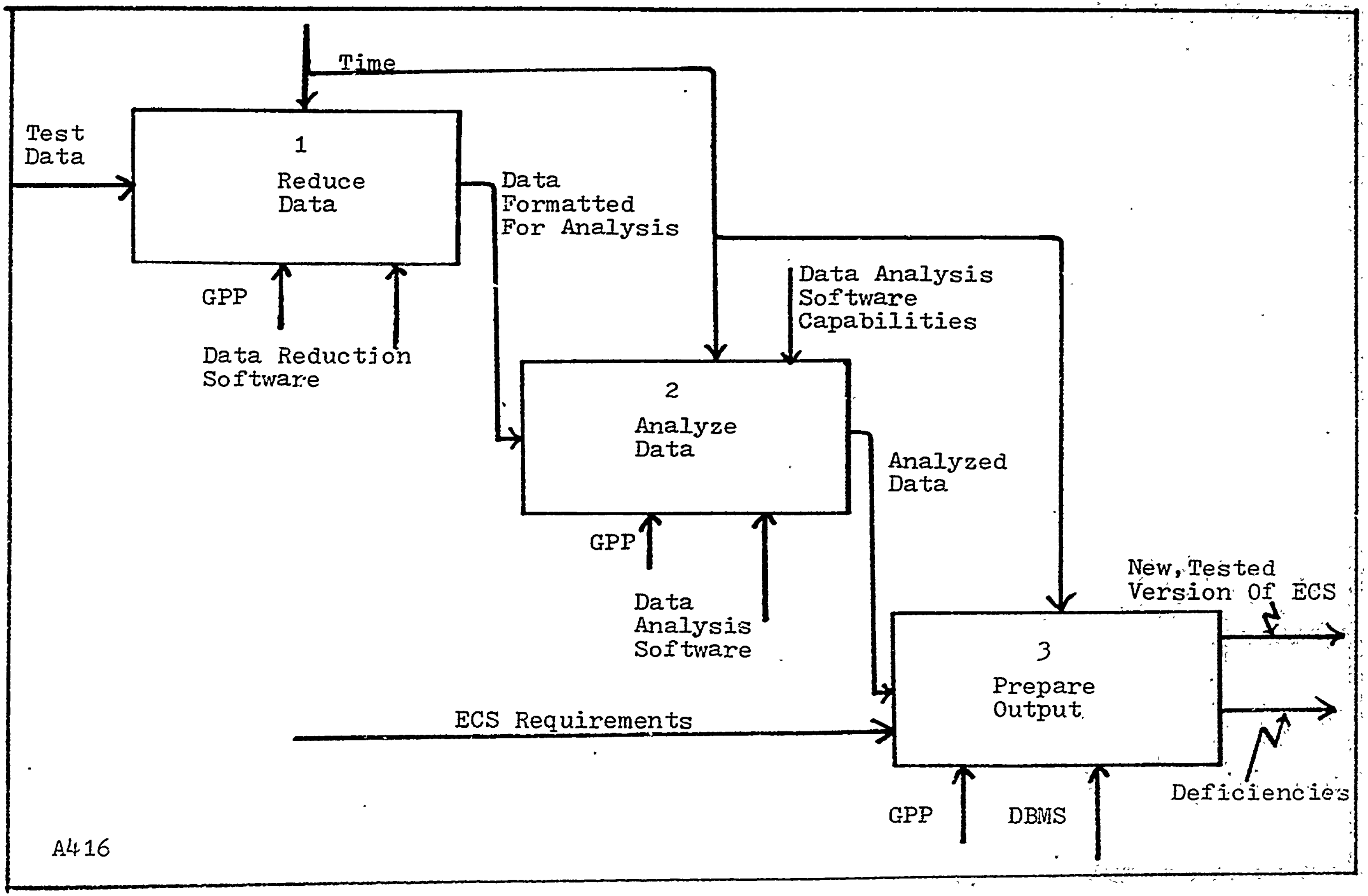


Figure 71. Analyze ECS Test Results

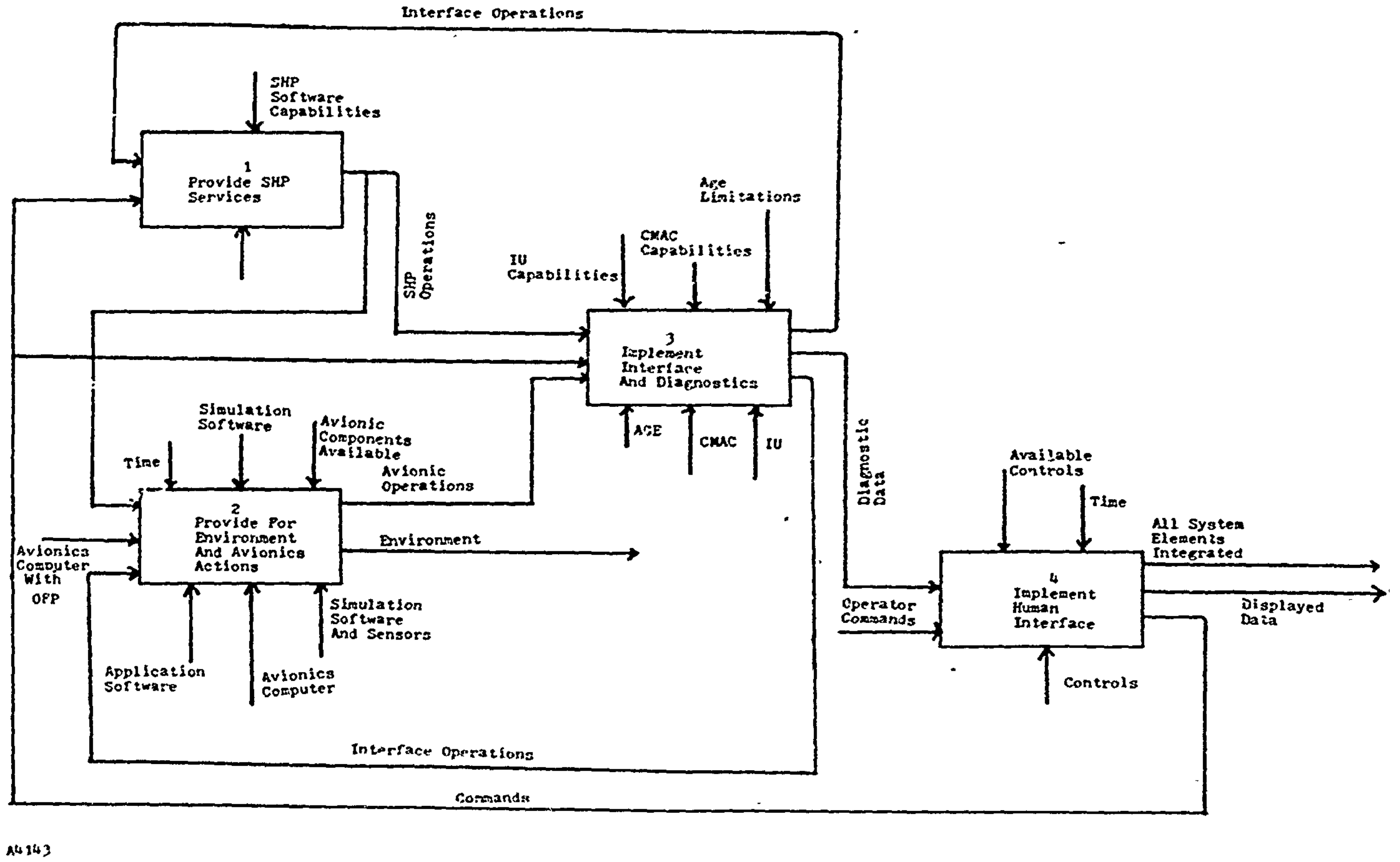


Figure 72. Implement Integration Test Bed

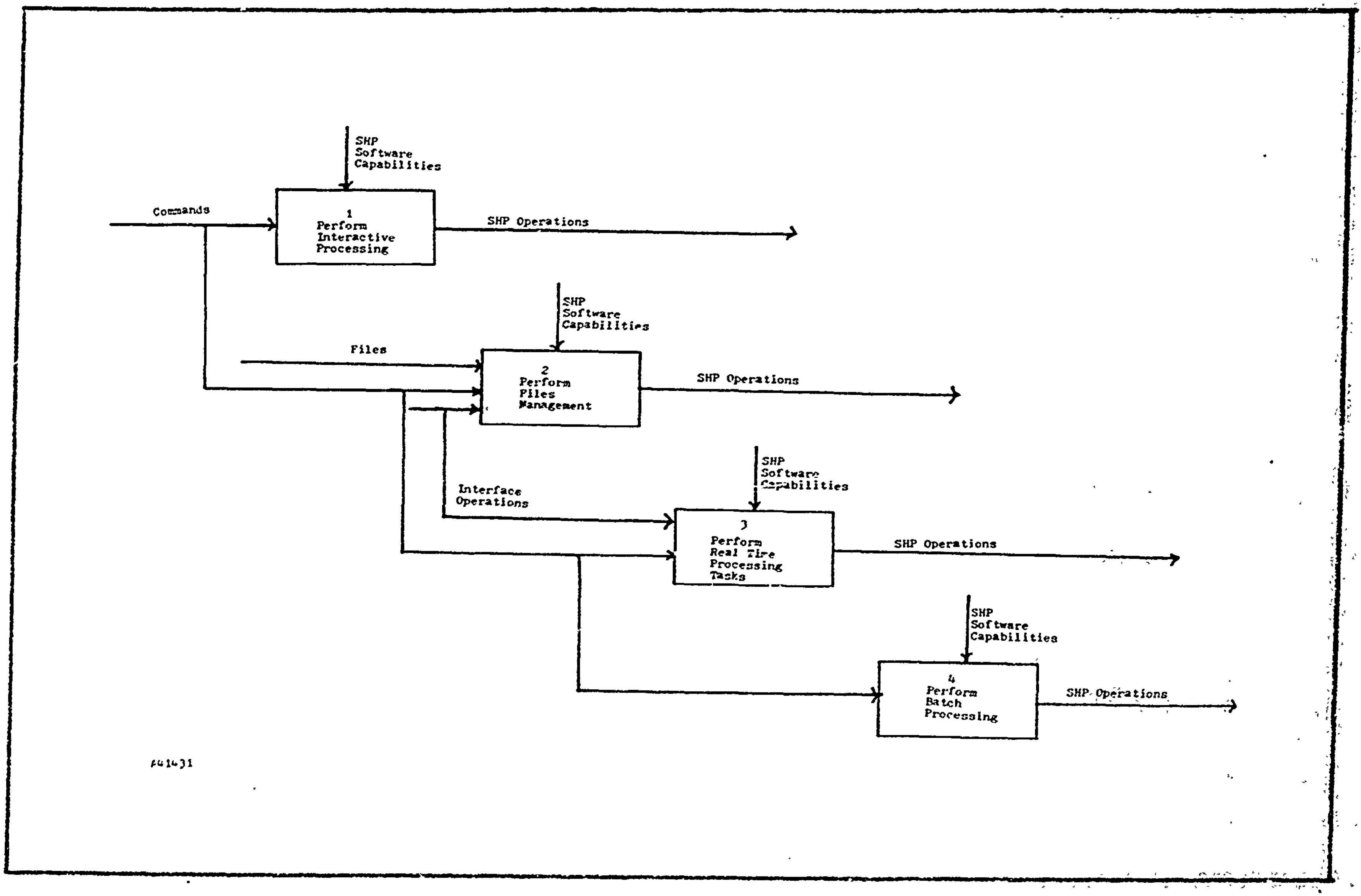


Figure 73. Provide SHP Services

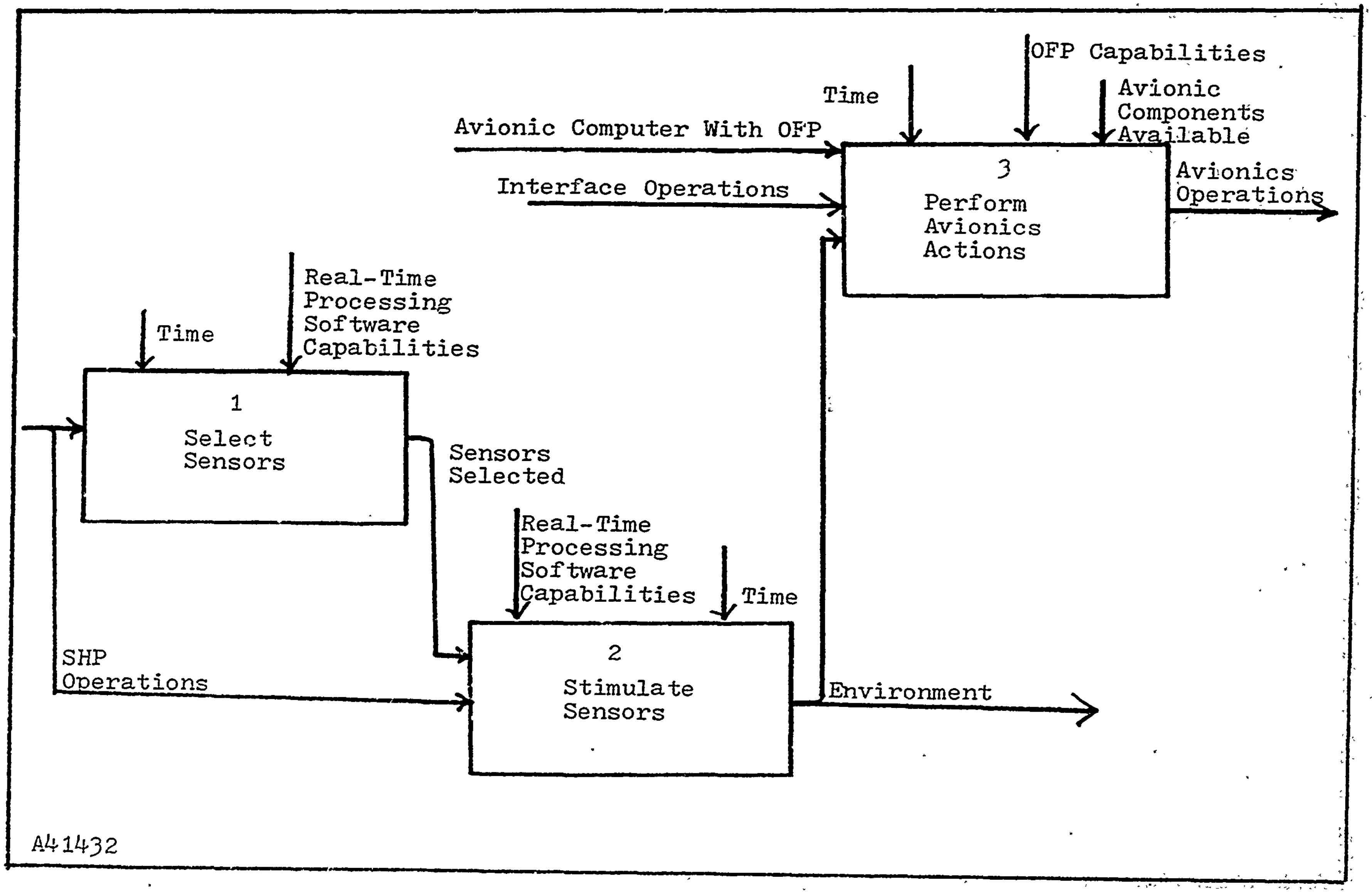


Figure 74. Provide Environment And Avionics Actions

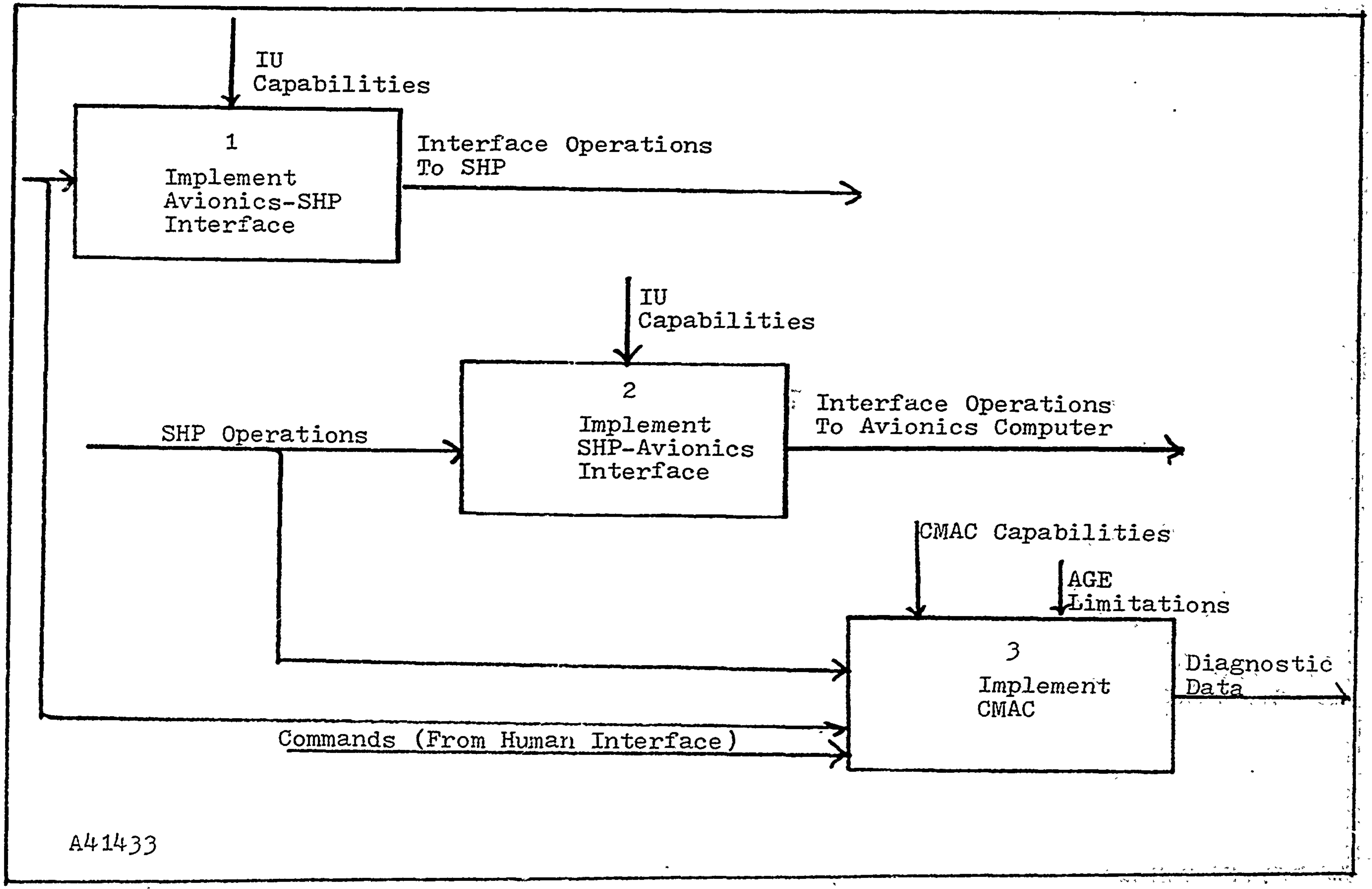
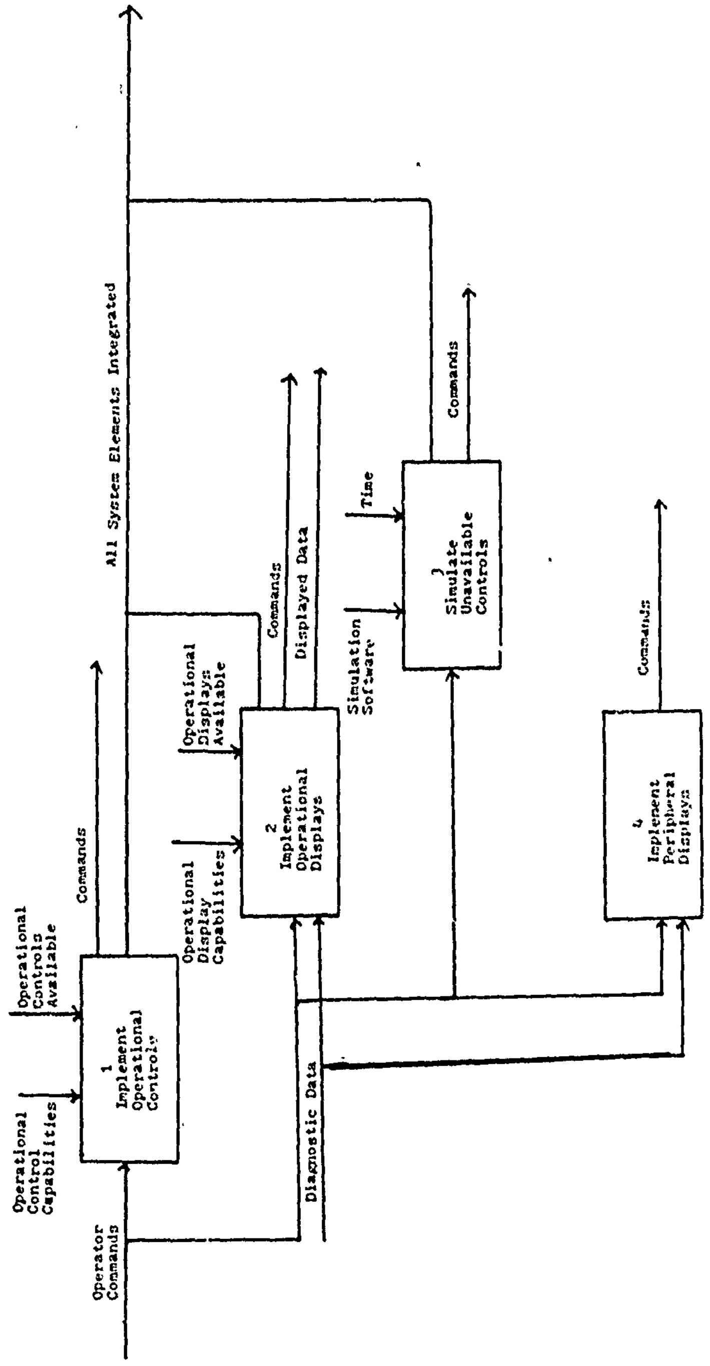


Figure 75. Implement Interface And Diagnostics



Operator Commands
Operational Control Capabilities
Operational Controls Available
1 Implement Operational Controls
Commands
All System Elements Integrated
Operational Display Capabilities
Operational Displays Available
2 Implement Operational Displays
Commands
Displayed Data
Diagnostic Data
Simulation Software
Time
3 Simulate Unavailable Controls
Commands
4 Implement Peripheral Displays
Commands
A41434

Figure 76. Implement Human Interface

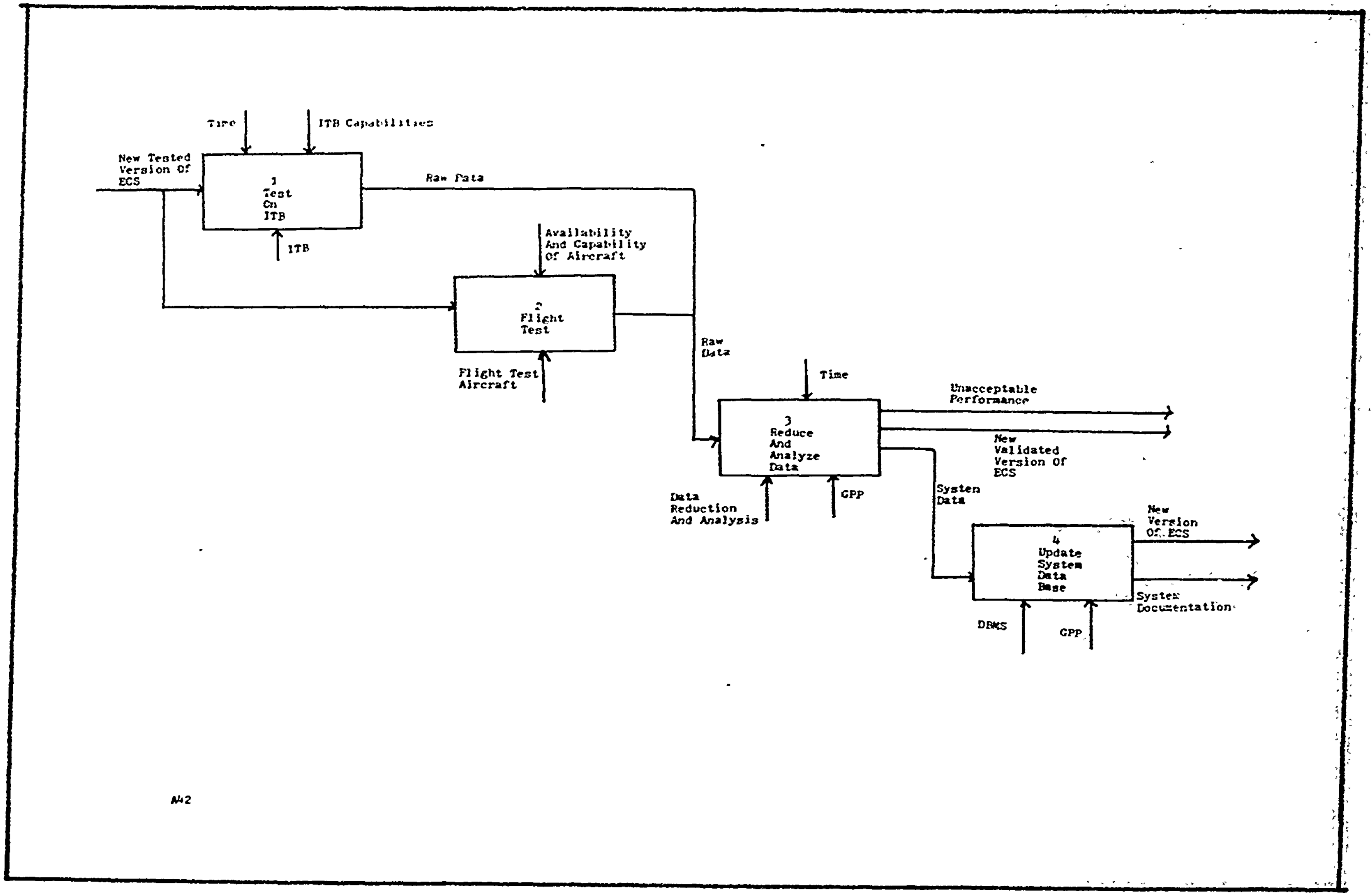


Figure 77. Accomplish System Acceptance

The mechanism by which the process is accomplished is the AISF.

Figure 60 shows the next level of detail, the four phases of the ECS change process with the Verification and Validation activity.

Analysis Phase

Figure 61 models the Analysis Phase of the ECS change process. The result of a successful Analysis Phase is a set of system requirements. It is hoped that these requirements are complete and valid. If not, the inadequacies will be detected in later phases, and portions of this phase will have to be reaccomplished to assure complete, valid system requirements for the ECS.

Design and Mechanization Phase

Figure 62 models the Design and Mechanization Phase which yields a detailed OFP design as well as a program test plan and test criteria.

Code and Unit Test Phase

The Code and Unit Test Phase of the ECS change process is modeled in Figure 63. The products of this phase are the tested, bug-free modules composing the OFP.

Test and Verification Phase

The model given in the remaining diagrams details the actions of the Test and Verification Phase. The portion of the model given in detail encompasses the Integration Test Bed implementation of the System Integration process. This is the process further examined in the next two chapters.

The end result of this phase, coupled with system validation, is the new, validated, version of the ECS deployed in the operational Air Force.

Verification and Validation

Though the Verification and Validation activity is modeled as a unique activity in Figure 60, it is, in fact, integral to all four phases. The reason it is illustrated separately in Figure 60 is to stress its importance in the ECS change process and to illustrate its interaction with all four phases. In reality the Verification and Validation process occurs during all four phases in the ECS change process.

Summary

Presented in this chapter is a complete SADT Actigram model for the AISF. In developing the model the same types of tradeoffs considered in Chapter III were examined. The model was kept as general as possible in order to facilitate the implementation of a relatively-standard AISF. As weapon systems become more standardized, so will their support facilities.

The system context is presented then decomposed into the four basic phases of the ECS change process. Each phase is, in turn, decomposed to the most basic level of detail showing all activities accomplished along with controls regulating the activity and mechanisms by which the activities are accomplished. Activities at any given level of detail provide the context for the next level of detail.

In the next chapter, an actual implementation of the "Integrate Systems" process of the Test and Verification Phase will be examined.

V. Design of KC-135 Hot Bench

Introduction

In the next two chapters a specific implementation of the Integration Test Bed (ITB) concept will be examined. A portion of the functional requirements discussed in Chapter III and modeled in Chapter IV is utilized and verified in an actual Air Force system. This chapter is intended to serve as a case study illustrating the actual application of the functional requirements of the "Implement Integration Test Bed" portion of the SADT model of Chapter IV. It deals with the design of the ITB from the functional requirements. The next chapter provides a Q-GERT implementation of them.

In this particular application, the ITB architecture is referred to as an Avionic Modernization Hot Bench (AMHB). The AMHB or Hot Bench is being designed and constructed by the Systems Engineering Avionics Facility (SEAFAC) division of the USAF System Command's Aeronautical System's Division (ASD). The KC-135 Hot Bench will serve as an Integration Test Bed for engineering evaluation and validation of the design and performance requirements for the modernized KC-135 avionics suite. Its primary objective will be to demonstrate an optimally designed and configured avionic suite that can be managed by a pilot and a copilot to accomplish KC-135 missions without a navigator. (Ref 19:1)

First, the purpose of the Hot Bench as well as SEAFAC's mission will be discussed. Then, the Hot Bench implementation will be examined relative to the model presented in the previous chapter. As each component of the Hot Bench is discussed it will be related to the functional requirements it fulfills in the SADT model. These relationships will be presented in tabular form throughout the chapter. Finally, the present level of implementation of the Hot Bench at SEAFAC will be discussed.

Background

Hot Bench Purpose. Due to the high cost of developing new systems and the budget constraints in recent years, the Air Force has been seeking ways to improve the performance capabilities and operating efficiencies of its strategic forces without incurring these high development costs. For this reason the Air Force is evaluating the feasibility of modernizing currently operational aircraft including the KC-135 aerial refueling tanker fleet. This modernization will include updates of existing avionic sensors and subsystems as well as the addition of new avionics subsytems for expanded mission capabilities. It is the desire of the Air Force to improve performance, lessen crew workload, and hold down the cost of modernizing th KC-135. These goals can best be accomplished through integrated avionics and centralized cockpit and display management techniques, all integrated into one system. This approach will yield life cycle cost savings resulting from mission and system performance optimizations possible with integrated avionics, as well as from reduced equipment costs arising from the consolidation of control and display functions. Crew workload can be reduced through more efficiently coupled avionics and fewer controls and displays. (Ref 31:4)

The Airlift Modernization System Program Office (SPO) has been directed to develop, specify, and demonstrate a system optimized avionics retrofit kit for the KC-135 fleet and to assess the possibility of performing the refueling mission without a navigator. The latter task is to be accomplished by the reduction of crew workload with an optimally configured retrofit of computers, sensors, controls, and displays. (Ref 31:4)

A critical engineering task within KC-135 modernization is the definition and specification of the avionic architecture and all hardware and computer software comprising the retrofit kit. The SPO has been directed to rely on in-house resources to the maximum extent possible in order to minimize costs. In complying with this direction, the SPO has authorized the in-house advanced development, construction, and operation of a Hot Bench representing a proposed KC-135 avionic system architecture. The Hot Bench will support an engineering evaluation of avionic system performance, cost, and reliability, emphasizing information processing and management, controls, and displays, and software. (Ref 31:4)

<u>SEAFAC's Mission</u>. The bulk of the engineering work for the KC-135 Hot Bench will be accomplished in-house by ASD engineers using the resources of the SEAFAC. SEAFAC is oriented toward engineering analysis, design, simulation, and evaluation of avionic architectures and equipment in a total system context. SEAFAC's specific objectives are to provide a Hot Bench capability to investigate alternative avionic

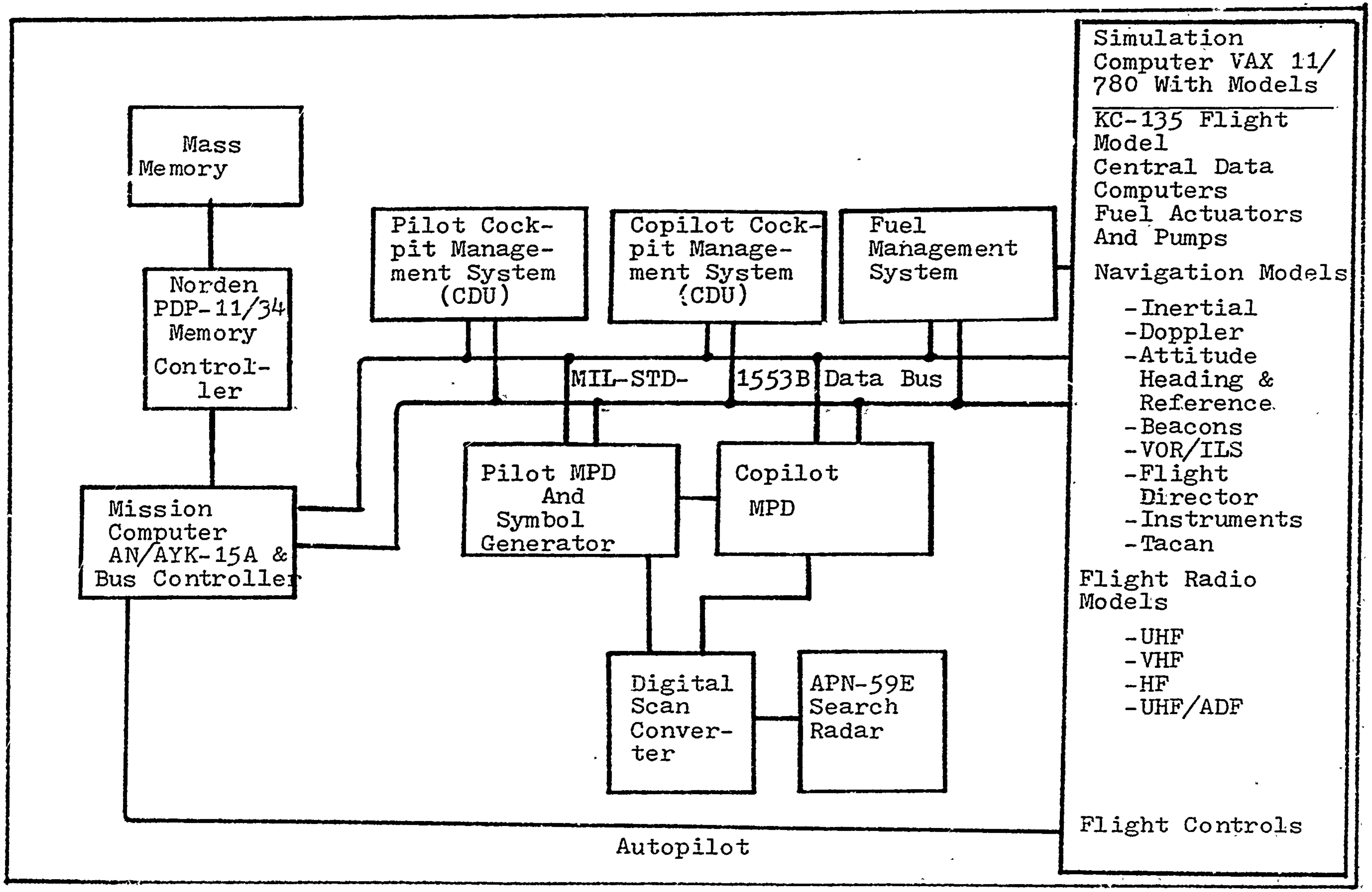


Figure 78. KC-135 AMHB

system designs in both hardware and software contexts, and to perform hardware and software verification. They also provide an ASD capability to define, evaluate, and specify avionic systems, and, to validate/verify compliance of hardware with avionic standards. Finally, SEAFAC provides a medium for transitioning laboratory and industry developed technology into the ASD engineering community. (Ref 31:5)

Hot Bench Design

The Hot Bench will consist of three major functional parts: the Modernization Avionics Suite (MAS), Environmental Systems Simulation (ESS), and the Hot Bench Monitor (HBM). The MAS in turn is composed of the Representative Avionics Subsystems (RAS) and Simulated Avionics Subsystems (SAS). The MAS is the fundamental core of the Hot Bench. The ESS provides support to the MAS allowing it to function in a ground environment. The HBM interfacing with both the MAS and ESS will provide the CMAC capability to view and collect data on the Hot Bench operation. (Ref 19:1,2)

Figure 78 depicts a block diagram of the Hot Bench which will consist of off-the-shelf, functionally representative, components of the major subsystems planned for the KC-135 avionics modernization including a MIL-STD-1750 mission computer (AN/AYK-15A) with mass memory controlled by an AN/UYK-42(V) (Norden PDP-11/34M) computer, a dual redundant MIL-STD-1553 Multiplex Data Bus, a Fuel Management Panel, Cockpit Management System (CMS), Control/Display Units (CDU), Multipurpose Displays (MPD), Tacan and radar altimeter, and a search radar

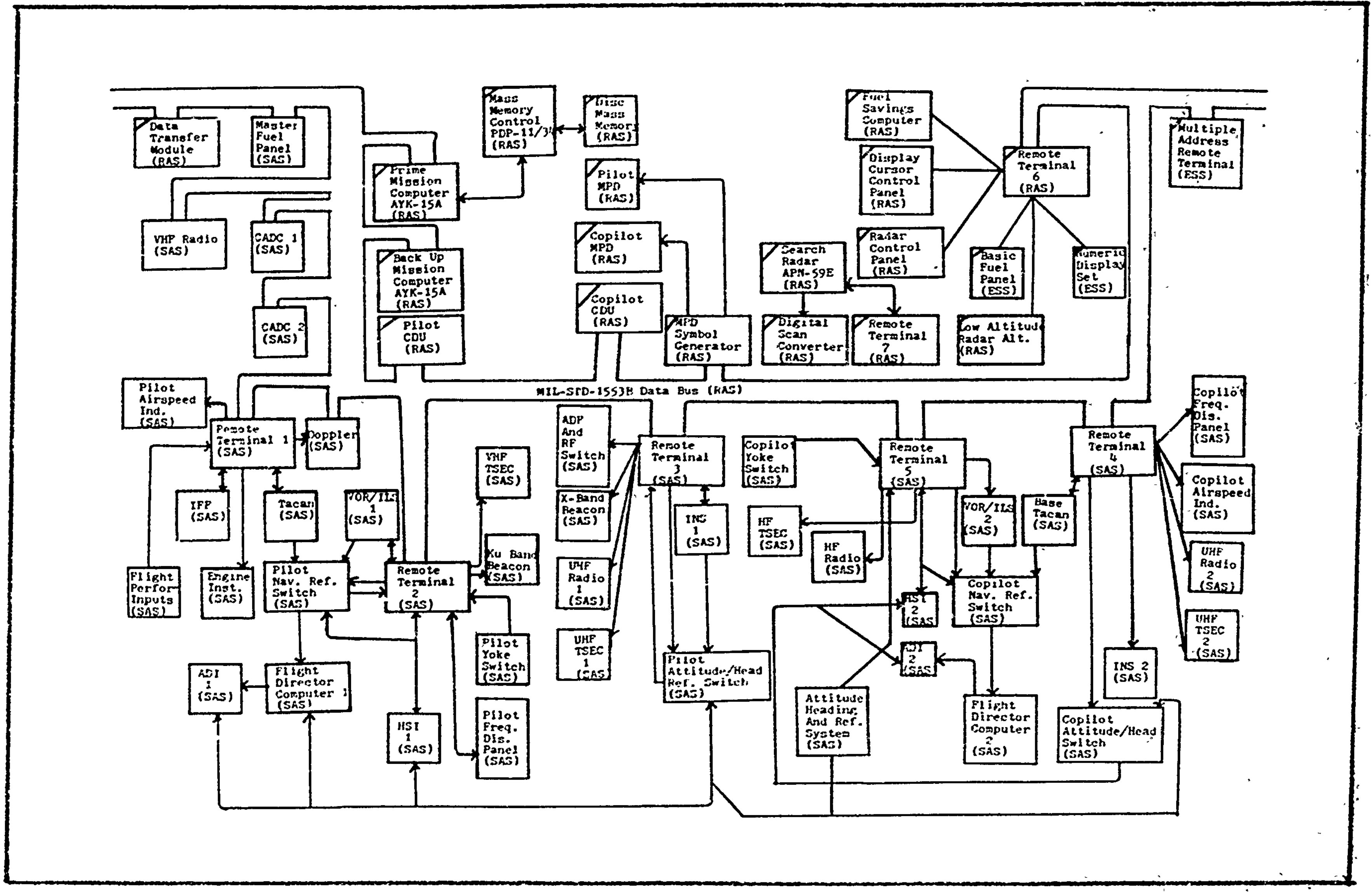

Data Transfer Module (RAS)
Master Fuel Panel (SAS)
Mass Memory Control PDP-11/34 (RAS)
Disc Mass Memory (RAS)
Prime Mission Computer AYK-15A (RAS)
VHF Radio (SAS)
CADC 1 (SAS)
CADC 2 (SAS)
Back Up Mission Computer AYK-15A (RAS)
Pilot CDU (RAS)
Pilot MPD (RAS)
Copilot MPD (RAS)
Copilot CDU (RAS)
MPD Symbol Generator (RAS)
Fuel Savings Computer (RAS)
Display Cursor Control Panel (RAS)
Search Radar APN-59E (RAS)
Radar Control Panel (RAS)
Digital Scan Converter (RAS)
Remote Terminal 7 (RAS)
Remote Terminal 6 (RAS)
Basic Fuel Panel (ESS)
Numeric Display Set (ESS)
Low Altitude Radar Alt. (RAS)
Multiple Address Remote Terminal (ESS)
MIL-STD-1553B Data Bus (RAS)
Pilot Airspeed Ind. (SAS)
Remote Terminal 1 (SAS)
Doppler (SAS)
IFF (SAS)
Tacan (SAS)
VOR/ILS 1 (SAS)
Flight Perfor Inputs (SAS)
Engine Inst. (SAS)
Pilot Nav. Ref. Switch (SAS)
Remote Terminal 2 (SAS)
VHF TSEC (SAS)
Xu Band Beacon (SAS)
ADI 1 (SAS)
Flight Director Computer 1 (SAS)
HSI 1 (SAS)
Pilot Yoke Switch (SAS)
Pilot Freq. Dis. Panel (SAS)
ADF And RF Switch (SAS)
X-Band Beacon (SAS)
UHF Radio 1 (SAS)
UHF TSEC 1 (SAS)
Remote Terminal 3 (SAS)
INS 1 (SAS)
Pilot Attitude/Head Ref. Switch (SAS)
Copilot Yoke Switch (SAS)
HF TSEC (SAS)
HF Radio (SAS)
Remote Terminal 5 (SAS)
VOR/ILS 2 (SAS)
Base Tacan (SAS)
Copilot Nav. Ref. Switch (SAS)
HSI 2 (SAS)
ADI 2 (SAS)
Attitude Heading And Ref. System (SAS)
Flight Director Computer 2 (SAS)
Remote Terminal 4 (SAS)
Copilot Freq. Dis. Panel (SAS)
Copilot Airspeed Ind. (SAS)
UHF Radio 2 (SAS)
UHF TSEC 2 (SAS)
INS 2 (SAS)
Copilot Attitude/Head Switch (SAS)

Figure 79. AMHB Components

(APN/59E) with digital scan converter. Computer models resident on a Simulation Computer (DEC VAX-11/780) will provide conventional avionic and flight control capabilities provided by other sensors and equipments such as radio and navigation aids. These models will simulate subsystem and sensor responses to crew control and operations as well as sensor performance characteristics and typical sensor interactions with the MIL-STD-1553 Data Bus. Sensor hardware and computer models will be exercised in real time by simplified receiver aircraft and radio navigational aid stations hosted on the VAX-11/780. The KC-135 flight simulation model will also reside in the VAX. A DEC PDP-11/55 computer system will permit on-line performance evaluation through management of performance monitoring interfaces and displays. (Ref 31:5)

Figure 79 shows, in more detail, all of the Hot Bench components. Included in each component block is an annotation indicating which functional portion of the Hot Bench that particular component belongs to (RAS,SAS,ESS). Components with a small triangular shaded region in the upper left-hand corner are actual hardware components, all others are software.

The Modernization Avionics Suite (MAS) consists of the RAS which is made up of off-the-shelf avionics components, and the SAS which is a group of simulation models which accomplish the remainder of the avionics functions. The MAS components are all interfaced via the MIL-STD-1553 Data Bus.

Representative Avionics Subsystems (RAS). The main hardware components of the RAS are the dual mission computers (AYK-15A) built to MIL-STD-1750 requirements on which the Operational

Flight Program (OFP) is run. The functions performed by the mission computer software are listed in Table 6. The second mission computer acts as a back-up for the primary mission computer.

Table 6

<u>Functions of Mission Computer Software</u>

Data Bus Management
Cockpit Management
Automatic Navigation Management
Automatic Position Computations
Aircraft Performance Management
Flight Plan Management
Aircraft Guidance
Rendezvous Management
Data Base Management
Special Features
MAS Status Monitor

The RAS also includes a Mass Memory accompanied by a Norden PDP-11/34M controller. Many of the actual display units to be used in the KC-135 are also included in the RAS including control display units, multipurpose displays, and a display cursor control panel. A search radar (APN-59E) which may be used for navigational, weather, or interrogation-receiver purposes is also included in the RAS. Accompanying the radar are a Multipurpose Display Symbol Generator which is interfaced to the search radar via the Radar Digital Scan Converter. The symbol generator is controlled via the data bus. Remote Terminals 6 and 7 are interface hardware components of the RAS which interface the data bus to other RAS components. A low Altitude Radar Altimeter provides the crew an indication of terrain clearance from 5000 feet. The remaining components of the RAS are the Fuel Savings Computer which computes minimum fuel consumption for a given flight plan, and the Data Transfer Module (DTM). The DTM is a small memory unit with a capacity of 8,192 16-bit words used for loading flight planning information into the avionics system and to provide a removable media for the storage of in-flight log data, crew notes,and system health data. (Ref 19:8-12)

Components of the RAS are involved in three of the four processes discussed in Figure 72 of Chapter IV, including: Implement Interface and Diagnostics, Provide Environment and Avionics Actions, and Implement Human Interface.

Environment and Avionics actions are provided by the mission computers and accompanying OFPs. Interface and Diagnostics are implemented through Remote Terminals 6 and 7 and through the Multiplex Data Bus. The Human Interface functions are accomplished by the display units and control panels. A breakdown of the RAS components and their functions relative to the SADT model is given in Table 7.

Table 7

RAS Components by Function

Funct.	Environment/ Avionics	Interface/ Diagnostics	Human Interface
Component	Mission Computers	Dig. Scan Converter	Control Display Units
	Mass Memory	Remote Term. 6	Multipurpose Displays
	Mass Memory Controller	Remote Term. 7	Symbol Gen.
	Search Radar	Multiplex Data Bus	Display Cursor Control Panel

Table 7 (Continued)

Funct.	Environment/ Avionics	Interface/ Diagnostics	Human Interface
Component	Low Alt. Radar Altimeter Fuel Savings Computer Data Transfer Module		Radar Control Panel

Simulated Avionic Subsystems (SAS). The SAS includes all of the simulated avionics subsystems running under the control of the ESS Executive Software and hosted on the VAX-11/780 computer. Subsystems included in this segment are those which are impossible or impractical to include as actual hardware in the Hot Bench. The SAS interacts with the RAS and ESS via the Multiplex Data Bus which interfaces the SAS at five Remote Terminals. These terminals are software simulation models of the type hardware remote terminals included in the RAS. They act as interface nodes tying the Multiplex Data Bus to the SAS simulations accomplishing navigational and communication functions.

The Central Air Data Computer (CADC) Models interface directly with the Multiplex Data Bus. These models provide outputs of altitude, mach number, mach rate, indicated airspeed, true airspeed, angle of attack, and free air temperature. (Ref 19:14)

Software models in the SAS perform many of the KC-135 navigational and rendezvous functions. Systems simulated in the SAS include: Inertial Navigation Systems (INS), Doppler

Radar, Tacan Receivers, Base Tacan, Identification Friend or Foe (IFF), Flight Director Computers, Lateral and Vertical Guidance, Altitude Heading and Reference, and APN-69 (X-Band) and APN-134 (Ku-Band) Radar Beacons.

Communications services provided by the SAS include simulation of the following radio functions: UHF, VHF, HF, UHF Tactical Secure Voice (TSEC), and Vertical Omnirange/Instrument Landing System (VOR/ILS). (Ref 19:17-18)

The last major subsystems simulated by the SAS are the flight controls. These models include the Master Fuel Panel, Engine Instruments Set, Flight Performance Input Set, Airspeed Indicators, Frequency Display Panels, Horizontal Situation Indicators (HSI), and various switches including Yoke Switches, Altitude/Heading and Reference Switches, Automatic Direction Finder (ADF), and Radio Frequency (RF) Switches. (Ref 19:19,20)

Represented in the SAS are the following three functions from the SADT ITB model presented in Chapter IV (Figure 72): Provide Environment and Avionics Actions, Implement Interface and Diagnostics, and Implement Human Interface. Environment and avionics actions not accomplished in the RAS and ESS are accomplished by the navigational, rendezvous, and communications simulations of the SAS. Interface functions are accomplished by the Remote Terminal simulations and the Multiplex Data Bus. Portions of the overall Human Interface are provided by the Flight Control simulations. Table 8 contains a breakdown of the SAS simulations by function.

Table 8

SAS Components by Function

Funct.	Environment/ Avionics	Interface/ Diagnostics	Human Interface
Component	CADC 1 CADC 2 Doppler Radar INS Tacan Receiver Base Tacan IFF Altitude Head- and Ref. Flight Director Comp. 1 Flight Director Comp. 2 X-Band Beacon Ku-Band Beacon UHF 1&2 VHF Radio HF Radio UHF TSEC 1&2 VHF TSEC HF TSEC VOR/ILS 1&2	Remote Terminal 1 Remote Terminal 2 Remote Terminal 3 Remote Terminal 4 Remote Terminal 5	Master Fuel Panel Pilot Nav. Ref. Switch Copilot Nav. Ref. Switch Engine Instruments Set ADF and RF Switch Pilot Attitude/ Heading Ref Switch Copilot Attitude/ Heading Ref. Switch Pilot Airspeed Indicator Copilot Airspeed Indicator Pilot Yoke Switch Copilot Yoke Switch Dual Remote Heading Slew Switch HSI 1&2 Pilot Frequency Display Panel Copilot Frequency Display Panel

Environmental Systems Simulation (ESS). The ESS supports the MAS and facilitates its operation in a ground facility. Elements simulated in the ESS include the KC-135 aircraft itself, aircraft fuel systems, terrain features, and some of the radio navigational aids. The ESS also provides overall control of the ESS and MAS components. In this capacity it will accomplish executive and control functions discussed in the SADT model of Chapter IV.

Simulation models included in the ESS are the Air Vehicle Model, Fuel System Simulation, Engine Model, Receiver Aircraft Model, Radar Altimeter Control Model, and Numeric Display Set Control software. As in the SAS these simulation models are hosted on the DEC VAX-11/780 computer. The Air Vehicle Model represents the performance of the KC-135 aircraft and will provide suitable dynamic outputs of the simulated aircraft inertial state to other Hot Bench components. (Ref 19:23) The Fuel System Simulation includes models of the Basic Fuel Panel, a Master Fuel Panel Model, Fuel Management System Model, and a Fuel System Model. The Engine Model will consist of a simple representation of the KC-135's engine performance including fuel consumption and subsequent aircraft mass reduction. Receiver aircraft such as the B-52 will be simulated by the Receiver Aircraft Model. ESS simulation models will also control the Radar Altimeter and Numeric Display sets.

Executive and control functions are accomplished in the ESS through the operator controls, operator displays, Master Executive software, Simulation Executive software, and various handlers.

Operator control of all aspects of Hot Bench operation, except the Hot Bench Monitor, will be accomplished through one of the dedicated DEC VT-100 CRT terminals. Operator commands, including Initialize, Run, Stop, Freeze, and Continue, will be used to start, run and halt the Hot Bench. The operator is also able to introduce fault conditions into the SAS and parts of the ESS using a Fault command. A second VT-100 will be dedicated to control panel simulation and will allow the operator to interact with simulated equipment which, in reality, would have a dedicated control panel. An interactive capability will allow the operator to, in effect, "throw switches" and "set dials". (Ref 19:21) The operator also gets information from the Hot Bench through a Numeric Display Set consisting of dot matrix display elements. Information output to the Numeric Display Set includes command airspeed from the mission computer, true airspeed of the simulated aircraft, command altitude from the mission computer, and true altitude of the simulated aircraft. A plasma dot matrix panel will be used to indicate the master time code generated by the VAX. The plasma display will be interfaced to the VAX via an RS-232 serial interface.(Ref 19:22)

A software element of the ESS that controls all other ESS software functions is the Master Executive. It also implements the interface between the Hot Bench Operator and the ESS. All simulation models of the ESS and SAS are controlled by the Simulation Executive Software. Control will be achieved through a time-slice resource sharing methodology

Table 9

ESS Components by Function

Funct.	Provide SHP Services	Environment/ Avionics	Interface/ Diagnostics	Human Interface
Component	Master Executive Simulation Executive	Ground Truth Services Ground Truth Handler Air Vehicle Model Fuel System Model Fuel Mgt. System Model Fuel System Simulation Engine Model Receiver Aircraft Model Radar Altimeter Control	1553 Bus Services 1553 Bus Handler PCL-11 Link Handler	Operator Controls Operator Displays Control Panel Simulation Basic Fuel Panel Numeric Display Set Master Time Code Display Control Head Services Control Head Handler Basic Fuel Panel Handler Master Fuel Panel Model Numeric Display Set Control

All simulation models will be called by the Simulation Executive as subroutines with no parameters. (Ref 19:22) Simulation services including Control Head Service, 1553 Bus Services, and Ground Truth Services are also provided by the Simulation Executive Software.

The last software elements of the ESS, called Handlers, chiefly handle specialized input/output for display processing.

All four of the major functions involved in the SADT model implementation of an Integration Test Bed are represented in the ESS. Environment and Avionics as well as Human Interface functions are provided by the simulation models. Simulation Host Processor Services are provided by the Master Executive Software and the Handlers. Human Interface actions are performed by the Operator Control hardware and software, and displays. A breakdown of the ESS components by functions, delineated in Chapter IV, Figure 72, is shown in Table 9.

Hot Bench Monitor (HBM). The HBM will provide on-line monitoring and analysis of Hot Bench operating characteristics and parameters. The HBM and Performance Monitor and Interface Unit (PMIU) together provide overall Computer Monitor and Control (CMAC) for the entire Hot Bench. That is, together they accomplish the functions described in Figure 75 of the SADT model. States and register contents of the mission computers are provided to the HBM by the PMIU hardware. The PMIU will also synchronize the execution of the OFP and the simulated subsystems resident on the VAX computer. Conversely, the simulation software resident in the VAX will also control the starting and stopping of the OFP in the mission computers via the PMIU. The PMIU will provide an interrupt to the VAX at the beginning of each execution cycle (31.25 milliseconds) by setting a trace point of the clock interrupt cycle of the OFP. The OFP execution will be controlled through the PMIU registers which are accessable on the VAX Unibus. (Ref 19:29)

The HBM will be hosted on the DEC PDP-11/55 computer and shall use VT-52 CRT terminals for display. One of the VT-52s will be used for operating the HBM and will facilitate the introduction of commands to control the HBM including snapshot capability. The snapshot capability allows the HBM to record the contents of all simulation/analysis displays and the values of all pertinent/global simulation parameters on magnetic mass storage. If used with the ESS freeze command, a snapshot will permit the recording of pertinent mission computer data. Other monitor commands will allow the operator to select, before and during simulation runs, those variables he wishes to have extracted from the data set for display and recording. (Ref 19:25)

The software which exercises control over the HBM is the HBM Executive. The Monitor Executive provides access control to the VT-52 terminals as well as initialization, scheduling, and control of all display tasks.

Level of Implementation

Presently, the different components of the Hot Bench are in different stages of development and testing. All hardware components of the RAS are at SEAFAC except the Digital Scan

Converter Unit for the Search Radar. The AYK-15A mission computers are interfaced to the Mass Memory and the OFP is in the Code and Unit Test Phase of development. he OFP or Mission Management Software (MMS) is being developed by the TRW Corporation. The Multipurpose Display Units and Control Display Units are in place and have been fully tested. RAS components tied to Remote Terminal 6 are being built, in-house, at SEAFAC and are at various levels of development.

All simulation models in the SAS are in the final stages of development and are receiving final testing at this time. The Master Executive and Simulation Executive Software of the ESS are complete and tested. When the SAS simulation models have been completely tested they will be integrated with the Simulation Executive. The HBM is still in the beginning stages of coding and its hardware interface to the VAX, the PCL has been installed and completely tested. The interface between the VAX and mission computers, the PMIU, is completely built but not yet tested. Completion of testing of all components and final system integration is expected to occur in December 1981.

Summary

Chapters III and IV dealt with the AISF, at the highest level of abstraction, as an entire system. Functional requirements for all phases and aspects of AISF operations were developed and modeled. Since an implementation of the entire AISF is beyond the scope of this investigation, the focus was narrowed, in this chapter, to cover in depth, one aspect of one phase of the AISF operation. Attention was focused from the overall OFP change cycle to the Test and Verification Phase of the cycle, and, specifically upon the Integration Test Bed aspect of that phase. The portion of the SADT model of Chapter IV covered in this chapter is contained in Figures 72 through 76.

An actual implementation of the Integration Test Bed (ITB) concept discussed in Chapters III and IV, an Avionics Modernization Hot Bench for the KC-135, was discussed. Justification for the Hot Bench was presented as well as a short discussion of the mission of SEAFAC, the facility responsible for designing and operating the Hot Bench.

In the design discussion of the Hot Bench, it was broken into its major functional components and each component further decomposed into its individual parts. The function of each component was discussed relative to overall Hot Bench operation. Next, the components were mapped to the functions presented in the ITB model of Chapter IV. Finally, the present level of implementation of the Hot Bench was discussed.

In the next chapter the top-down approach incorporated in this investigation will be iterated to the lowest level of detail. Chapter VI focuses on a portion of the Hot Bench discussed here. A predictive model of the Hot Bench real time simulation software will be presented.

VI. Q-GERT Implementation of KC-135 Hot Bench Real Time Simulation Software

Introduction

In this chapter a portion of the design modeled in Chapter IV and described in Chapter V will be simulated using the Q-GERT Modeling and Analysis Language. A short tutorial on Q-GERT is presented in Appendix D. The majority of the simulation focuses on the executive and control software which makes up a large part of the ESS, however, the software simulation models of the ESS and SAS also play a major role.

Figure 80 depicts a block diagram of the Hot Bench showing major hardware, software, and interface components. Component details were discussed in Chapter V. All major components communicate via the Multiplex Data Bus. The mission computers interface with the Simulation Host Processor (VAX) through the Performance Monitor and Interface Unit Hardware. The Hot Bench Monitor (HBM) interfaces with the SHP through the Parallel Communications Link (PCL) hardware. The major component modeled in this simulation is the SHP and its associated software, as well as the two interface units (PCL,PMIU) previously discussed.

The actions of the PCL, PMIU and the five executive programs running on the SHP are modeled. Stored in the memory of the SHP are the following five executive programs: Master Executive, Simulation Executive, Display, Control Head, and the PCL Software which carries Monitor signals. Accompanying this executive and control software are various simulation

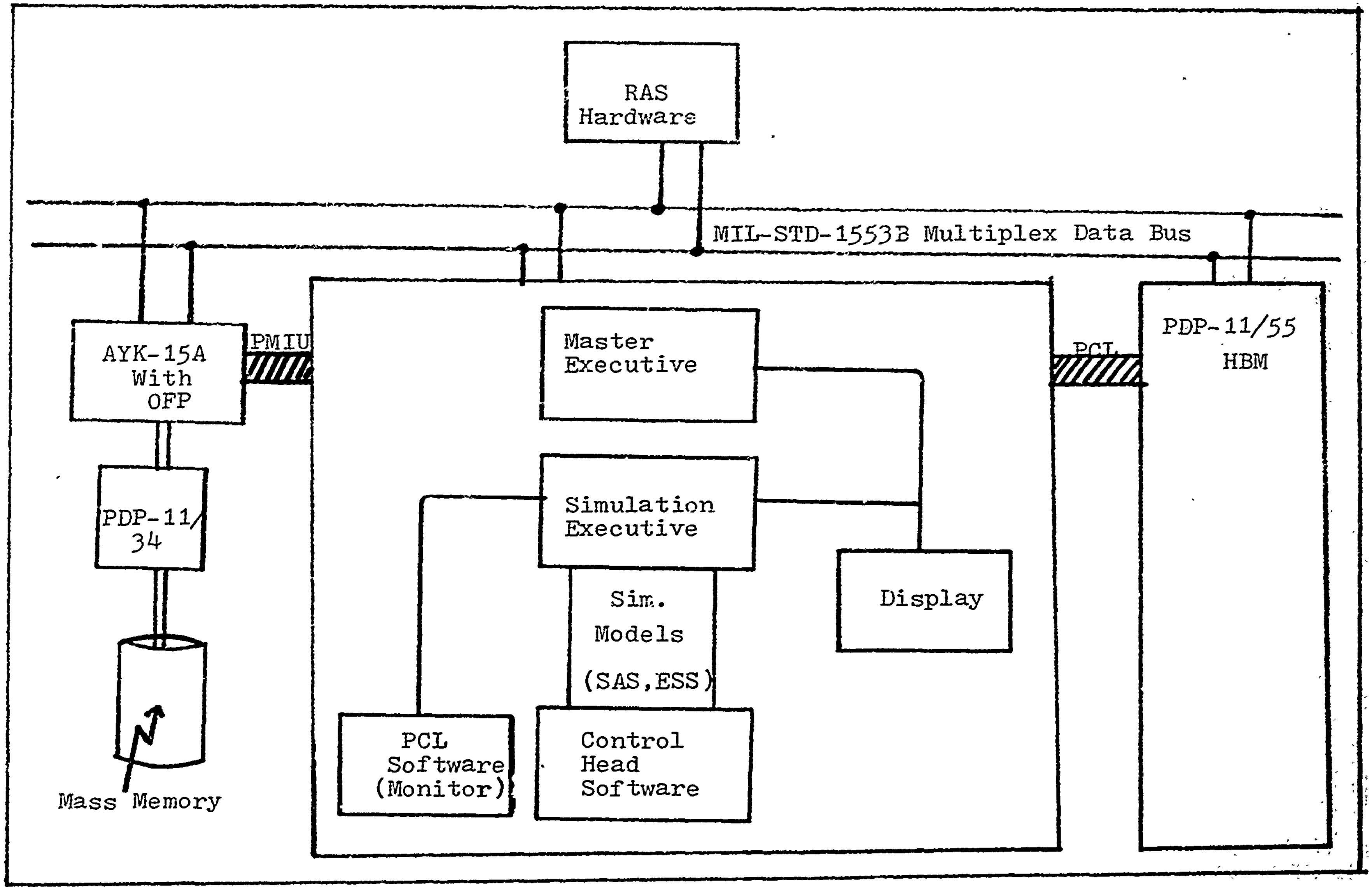


Figure 80. Hot Bench Block Diagram

Table 10

Input/Output Signals to VAX For KC-135 Hot Bench

Signal	Executive Program Called	Fuction
PMIU Signals	Simulation Executive	Call the simulation Executive which, in turn, calls the models to be run to simulate the environment.
Operator Commands	Master Executive	Commands issued by the Hot Bench operator which actually control all elements of the Hot Bench. Commands include Start, Stop, Freeze, Etc...
Fault Signals	Control Head	Establish system controls such as those normally input by the pilot via the yoke.
Display Signals	Display	Provide inputs to the display simulation which controls numeric displays for the Hot Bench.
Altimeter Signals	Display	Flight Level Inputs
Clock Signals	Display	Simulation time to tenth of second
Monitor Signals	Monitor	Makes Bus data available to display.

models making up the SAS and parts of the ESS. All five of these executive programs as well as the other utility programs, such as the Terminal Driver, compete for the SHP's single Central Processing Unit (CPU).

In a typical "flight" of the Hot Bench, the mission computers will accomplish such functions as cockpit management, flight planning, navigation functions, and fuel management. The SAS and ESS software simulation models, which are, in fact FORTRAN programs, provide sensor information to the flight computers. They will provide the "environment" for the flight computers to "fly" in. The SHP is responsible for overall Hot Bench control, control of data displays, and a means for running the real time simulation. It will accomplish these functions using the five executive programs loaded in its core memory and called by certain signal inputs across the Multiplex Data Bus, PCL, and PMIU. Table 10 briefly outlines the signals as they apply to the simulation.

In this chapter the SADT model of Chapter IV will be examined in the most detail. This is the lowest level of the top-down structure which dictates the structure of the investigation. The functional requirements diagrammed in Figures 72 through 76 of the SADT model are implemented here. The functional characteristics of the operation of the Hot Bench software on the VAX are modeled in the Q-GERT language.

Though the simulation could have been accomplished in a high order language such as PASCAL, Q-GERT was chosen because it is specifically intended for modeling and analysis. Its structure and statistics collection capabilities make it ideal for modeling the Hot Bench. Q-GERT is especially useful for modeling large systems because it allows the analyst to conduct experiments on a computer model and to examine the system being simulated under a given set of assumptions, conditions, and for a particular parametric model. Q-GERT allows the analyst to control and measure variables while studying all parts of a fully integrated system.

The purpose of the simulation is to determine critical design factors for the Hot Bench and to provide a predictive model to SEAFAC personnel. The model will enable them to predict how the final product will perform, and will verify that the present design for the Hot Bench is sound, or, verify that the design is not sound and provide alternatives to the present model. The final goal to be accomplished in this chapter is to illustrate the use and value of a specific software analysis and design tool (Q-GERT) in hot benching and the overall AISF environment.

Customer-Analyst Coordination

This simulation was accomplished to provide a meaningful product to SEAFAC and to aid in development of a major project. Meeting the latter objective required much close coordination between the analyst/modeler and SEAFAC. This coordination involved four interviews of a duration from two to four hours each, involving Air Force and civilian hardware and software experts. Initially, the analyst was presented with the overall scenario from which the model was designed. Once the model was designed, it was presented to SEAFAC for verification. Any inaccuracies were corrected and the model was ready to be run using the Q-GERT modeling and analysis tool. Results from initial runs were discussed with SEAFAC and experimentation goals determined.

Successful accomplishment of the project required much close coordination between those at SEAFAC and the analyst. Of prime importance was a mutual understanding of the criterion for success, and the accuracy of the Q-GERT model in depicting the actions of the VAX, and its software, in the hot bench scenario.

SEAFAC personnel were enthused at the prospect of having an accurate, predictive model of their hot bench and there are plans to embellish the model, to further incorporate other hardware and software facilities of the system, in the future.

Assumptions and Criteria

Assumptions. It is assumed that the Hot Bench accurately simulates the flight of the KC-135 aircraft, and, that the hardware and software designed, or being designed, will be sufficient for this purpose. The following specific assumptions apply:

1. The four megabytes of core memory for the VAX computer by far exceed the amount needed to contain the five executive programs and various models used in the Hot Bench. Therefore, core memory will be considered infinite for the purpose of this project.

2. One CPU is the maximum that will be available for running the Hot Bench; addition of a second CPU will not be considered feasible.

3. Unibus memory (64K) is so fast that it cannot be saturated during Hot Bench operation, therefore, it will be considered infinite.

4. All service times in CPU, PCL, and consoles are accurately modeled with a non-shifting Beta Pert distribution.

5. Operator commands, fault signal, and control head signal interrarrival times are accurately modeled with non-shifting Exponential distributions.

6. A high level of confidence in the accuracy of the interarrival rates is assumed.

7. A low level of confidence in the accuracy of the service times is assumed. Tests must be conducted to determine maximum allowable service times for a viable system.

8. Operator commands, fault signals, control head signals, and display signals all require the same amount of CPU time to run.

9. Altimeter and Clock signals both require the same amount of time to run on the CPU.

10. A simulation run time of 100 seconds will allow for exercising all aspects of the Hot Bench operation.

11. Thrashing programs into and out of the CPU is an undesirable characteristic, therefore, in Experiment Four, interrupts are not permitted.

12. In calculating PMIU service times on the CPU, the average number of environment models run in a given simulation is four, and the average run time per model is 1.5 milliseconds.

13. Computer reliability is not a factor in this model.

14. The Terminal Driver will always be available, before the next transaction needs it, therefore, it is not a critical resource and will not be represented in the network as such.

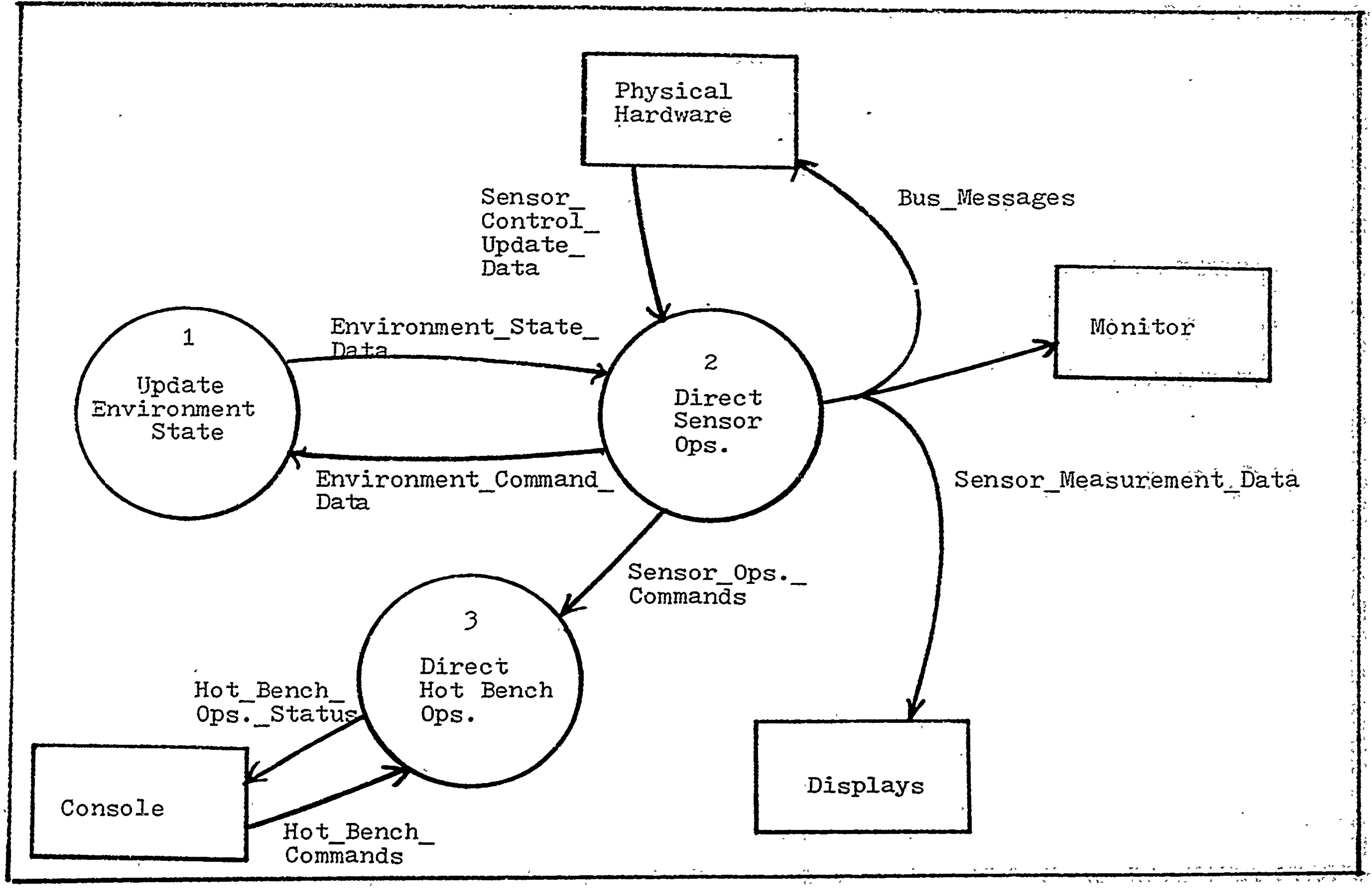


Figure 81. DFD Of VAX Role In Hot Bench Control

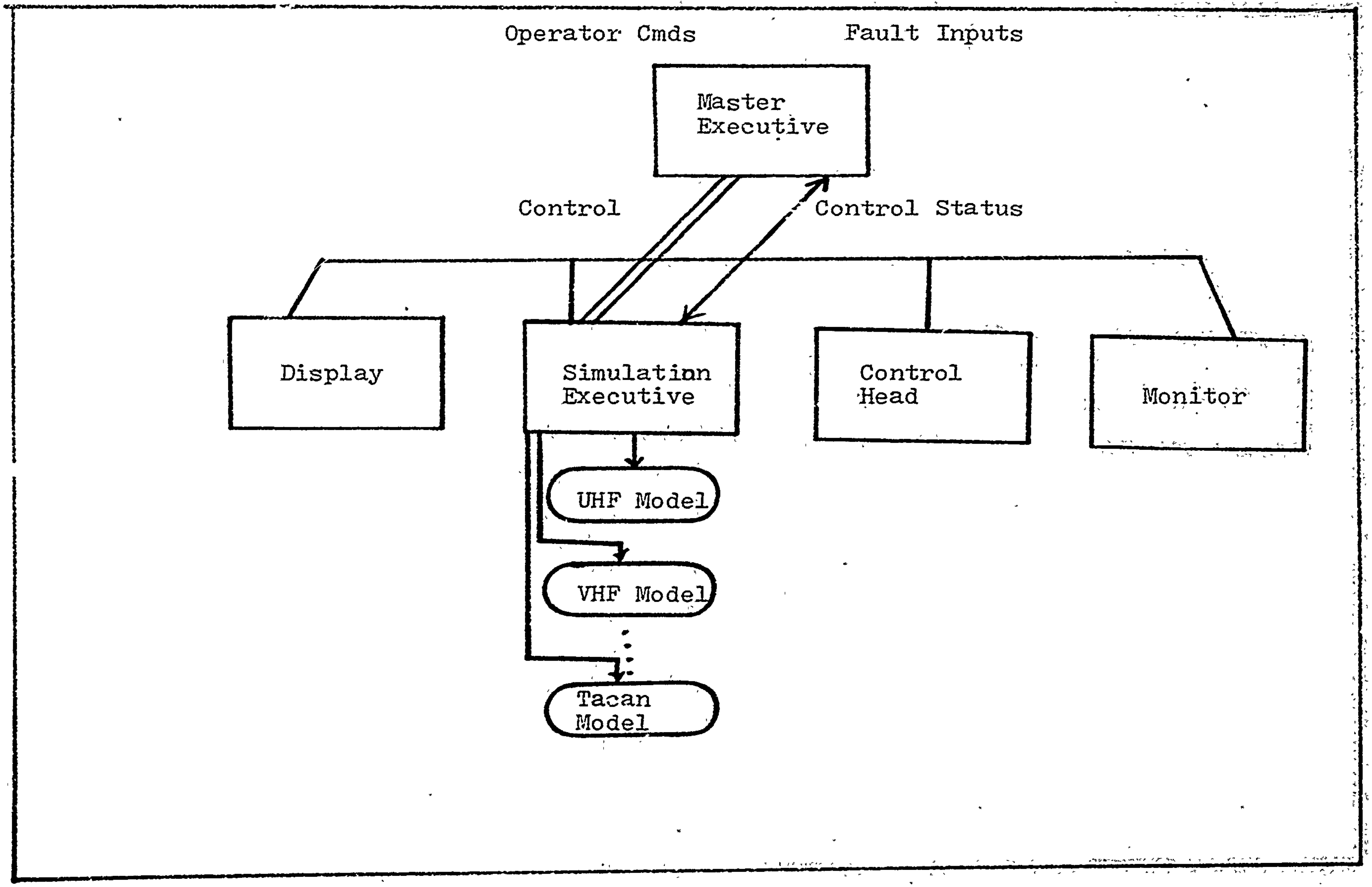


Figure 82. Hot Bench Executive Software

Criterion. The system will be deemed viable if, and only if, all signals spend less time in the system than their interarrival rates. That is, for all signals: TIS < IAT, where TIS= Time In System for a transaction (signal) and IAT=Interarrival Time for a transaction. Critical resources are those which are required to assure system viability. They are the CPU, PCL, various consoles, and the five executive programs in VAX core memory.

Design Experimentation and Results

Knowledge of the operation and design of the KC-135 Hot Bench was gained through extensive interviews with SEAFAC personnel. Attention was focused on the role played by the VAX in controlling the Hot Bench. The first milestone in developing a design was determinimg if, in fact, the role of the VAX in controlling the Hot Bench could be represented in a form suitable for Q-GERT analysis. Figure 81 depicts a DFD representation of the role of the VAX in the overall Hot Bench scenario. The two bubbles, "Direct Hot Bench OPs" and "Direct Sensor OPs", represent those activities performed by the VAX. Other bubbles and boxes in the diagram represent consoles, displays, hardware, and the environment state. Figure 82 depicts the calling sequence of the five executive programs which are stored in the VAX core memory and contend for the single CPU. These structures, combined with a general knowledge of the capabilities and resources of the VAX yielded the schematic diagram shown in Figure 83. This schematic illustrates the inputs, five programs in core, and flow through the computer.

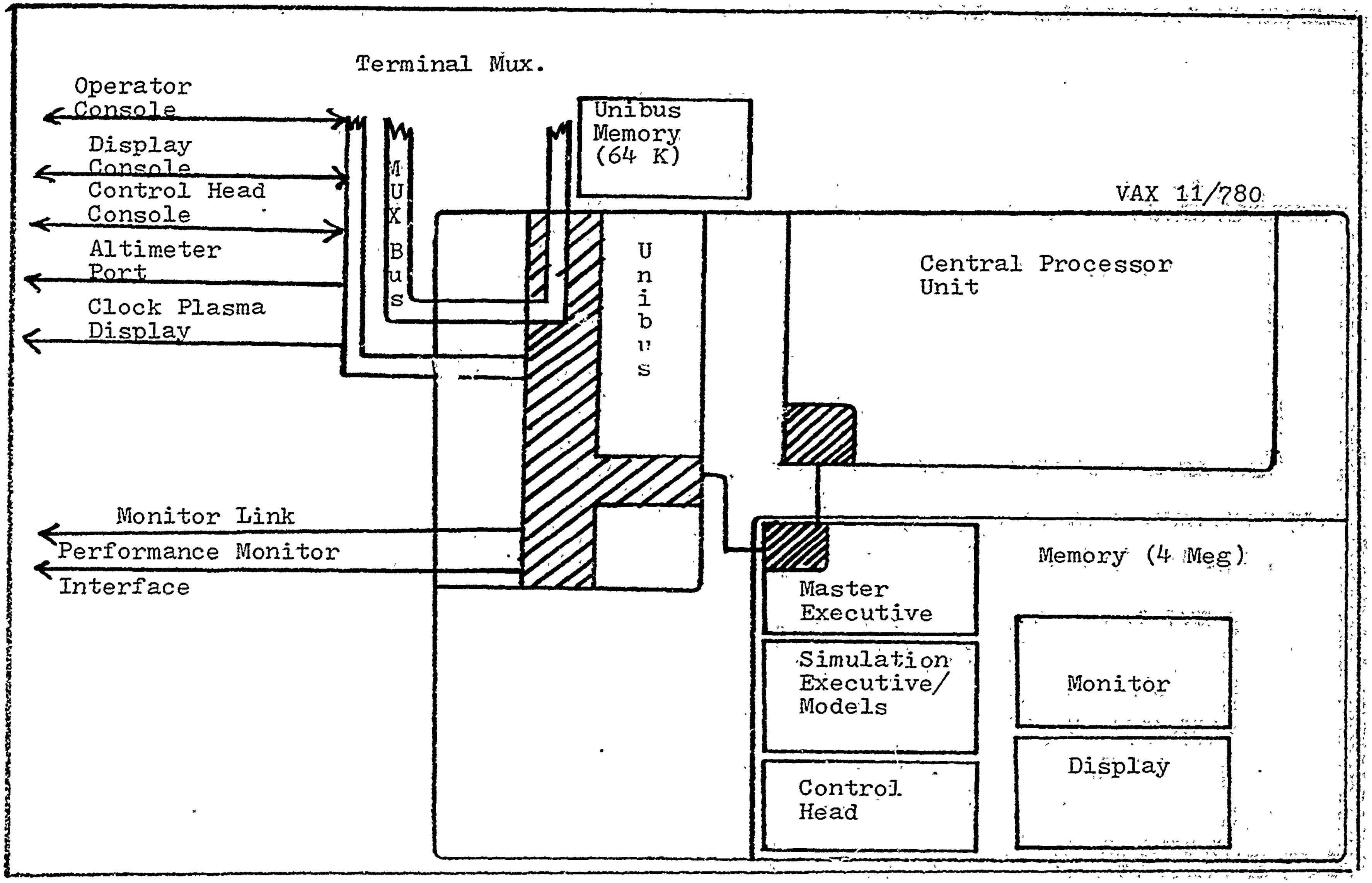


Figure 83. VAX Schematic

From the schematic the Q-GERT model was created. First, the critical resources, and points of contention for them, were determined. The signals to be included in the model were determined and assumptions were made concerning candidate transactions and resources for the network.

The next step involved designing and implementing the actual Q-GERT model. The complete Q-GERT network is contained in Appendix E. A total of twelve resources were identified and eight signals were identified as transactions. Each of the eight signals enters the network through a source node at a given interarrival rate. The interarrival rates were established as functional requirements for the Hot Bench and were supplied by SEAFAC. After the signals enter the network, they enter queues for the five executive programs stored in memory. The five programs are allocated and they join a FIFO queue to await processing on the CPU. The CPU is allocated and each of the five executive programs is run for a given service time. The service time to run a given executive program on the CPU is a function of the type of signal the program is simulating. The service times are selected by means of a conditional-take-first node checking attribute one, the identity attribute for each type of transaction.

After an executive program runs on the CPU (transaction processed by CPU), it enters a network of free nodes. First, the CPU is freed and the nodes, where it is allocated to the Terminal Driver, and, five executive programs, are polled respectively. This arrangement gives the Terminal Driver priority over the five executive programs for CPU allocation; based upon the relatively small time slice required for the Terminal Driver to run on the CPU. After the CPU is freed, the executive program used by the transaction is then freed. At this point, the PMIU signals leave the network and statistics on their TIS are collected. Monitor signals enter a queue for the PCL which is allocated, the Monitor signal processed, then the PCL freed. The Monitor signals then exit the network through a statistics node.

The six remaining signals: Operator Commands, Fault Signals, Control Head Signals, Display Signals, Altimeter Signals, and Clock Pulses proceed to a queue where they once again await allocation of the CPU, this time, so that the terminal driver may be processed for their output. Following the running of the terminal driver on the CPU, the CPU is once again released and the transactions are routed through a conditional-take-first node to appropriate queues to await allocation of their respective consoles and displays for output. These output devices are then allocated. Subsequent service times represent the time to output given signals. Next, the output devices are freed and statistics on the TIS for each transaction is recorded at statistics nodes.

The parametric model was developed based upon certain diagnostic and report generating programs running on the VAX. The times were mutually determined by SEAFAC personnel and the analyst, with the understanding that they were, at best, estimates. One of the important goals of the simulation was to determine what maximum allowable parameters would be, based upon the pre-defined criterion for viability.

Once the model was complete, it was transformed into a Q-GERT program. The original program used in Experiment One is contained in Appendix E. Using this model and program, with variations, five experiments were conducted involving a total of 59 configurations for the system. The configurations, described by experiment, are shown in Appendix E.

Table 11

Experiments On Q-GERT Model

Experiment Number	Description
Experiment 1	Run and test initial parametric model.
Experiment 2	Run corrected parametric model and increase simulation time to 100 seconds.
Experiment 3	Check for upper limits of the parametric model.
Experiment 4	Prioritize executive programs.
Experiment 5	Test effect of adding more executive software.

Experiment One consisted of the initial run of the model in order to determine if the original system, with the original parametric model, was viable. An initial ten second run showed the system to be totally unviable; the TIS for Altimeter signals was 2.2830 seconds and the maximum allowable TIS for this signal is .125 seconds. Results of Experiment One vis-a-vis the system viability criterion are shown in Table 12.

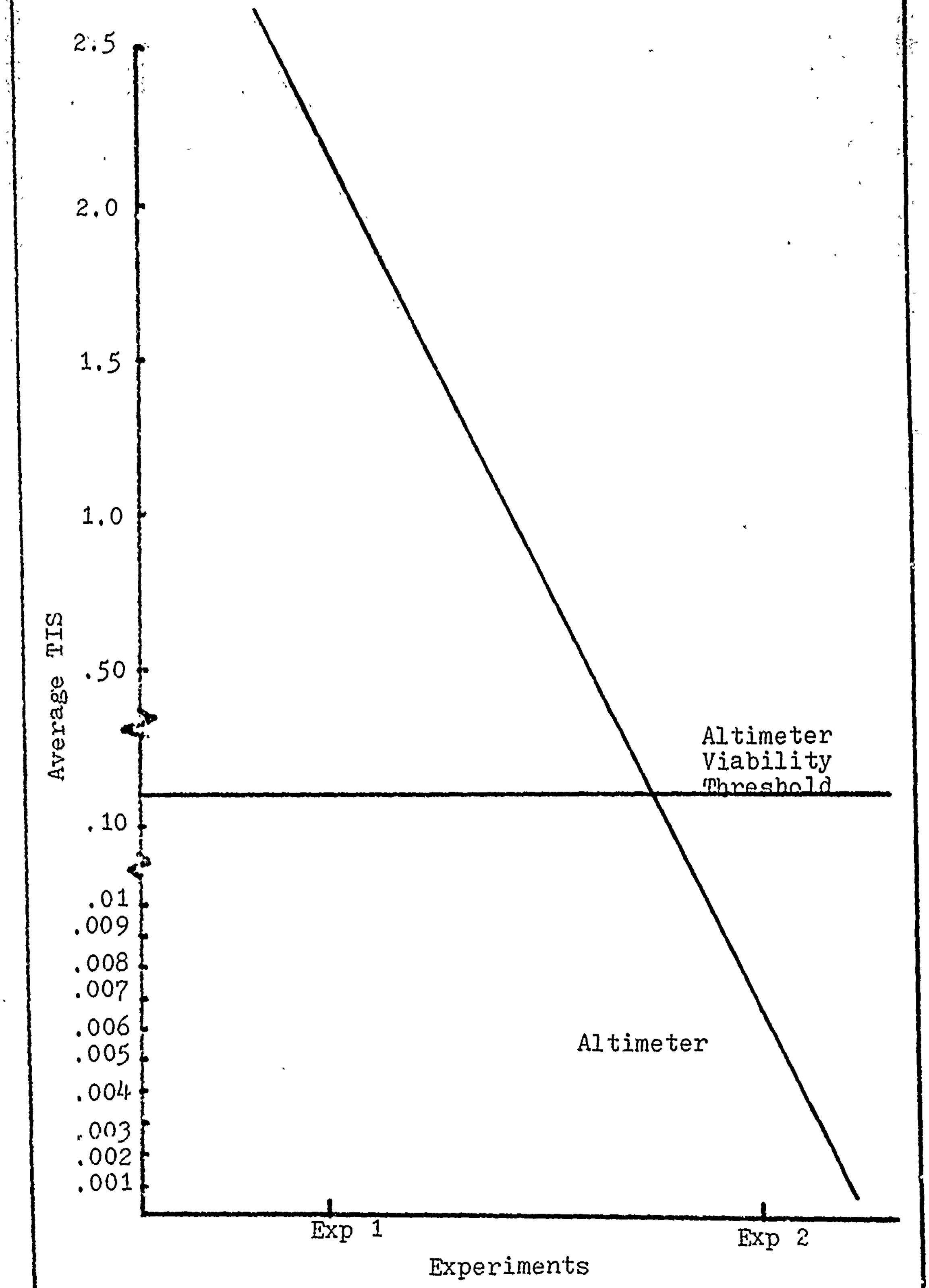


Figure 34. Results-Experiments One And Two

Table 12

Results of Experiment One

TIS

Signal	Exp-1	Viability Criterion
Clock	.2202	1.0
Altimeter	2.2830	0.125
Display	.5298	1.0
Control Head	.4101	120.0
Operator Cmds.	.6757	300.0
Monitor	.0082	0.031
PMIU	.0062	0.31

Investigation determined that a serious bottleneck was occurring at Node 60 in the network, where the Altimeter signals queued for output to the console. Apparently, the service time to output the Altimeter signal was too high. These conclusions were presented to SEAFAC and a recomputation of the output time accomplished. Based upon a maximum altitude of five ASCII characters, at 50 bits per second per character and an output rate of 9600 Baud, a new constant altimeter output time of .0052 seconds was computed. A new clock output time of .01 seconds was also computed in the same manner.

In Experiment Two these values were corrected in the program and the run escalated to 100 seconds, to achieve a more comprehensive simulation. The system was now viable as shown graphically in Figure 84. With the change in the parametric model a dramatic drop of more than two orders of magnitude was observed for the TIS of the Altimeter signals. The point of emphasis for Experiments One and Two is that the service times on the CPU and output devices have a profound effect upon system performance. They are obviously critical factors in system viability.

The system was now viable and the remainder of the analysis was devoted to examining different architectures and their effects upon system performance.

In Experiment Three an upper limit for the service time values in the parametric model was sought. The approach taken was to increase each service time on the CPU, holding the others constant at the levels determined for the original parametric model, until the system became unviable for any of the eight input signals. The experiment was conducted in this manner in order to find the maximum allowable CPU run time for each signal in the system. The reason that service times were increased until only the first of the eight signals became unviable was because, according to the original viability criterion, the system is deemed viable only when all signal TISs are less than IATs.

The results of Experiment Three showed that the original parametric model could be increased dramatically and the system would remain viable. The most sensitive signal to increases in all CPU run times was the Monitor signal. Results showed that, as a minimum, the PMIU run time on the CPU could be increased two and a half times and, as a maximum, the terminal driver run time on the CPU could be increased 350 times with the system remaining within the Monitor viability threshold of 0.031 seconds. These results are depicted in Appendix E. Each graph shows the effect upon the most sensitive input (Monitor) of increasing the various CPU run times. The curves remain relatively flat until the CPU run time approaches the value which will take it over the system viability threshold. At this point, the curve takes on the fast-rising characteristic of the Exponential Function. It should be noted that due to the radical range of the average TIS over the ranges of average CPU times tested, the values for average TIS are not necessarily linear.

The major results observed in Experiment Three are threefold: 1) Monitor is the most sensitive signal to the CPU run time, 2) the CPU run time values have a profound effect upon system performance and should be considered critical for system viability, and 3) the values determined in the parametric model can be increased greatly with the system remaining viable. The last conclusion should be of major interest to SEAFAC. Complete statistics from which the graphs were derived are contained in Appendix E.

In Experiment Four, the effect of prioritizing the executive programs for running on the CPU was observed. A prioritizing scheme more accurately represents the actions of the VAX operating system and CPU. Three prioritization schemes were observed, in order to determine which, if any, would improve system performance. The three schemes determined, based upon the relative importance of the executive programs to overall system performance are shown in Table 13.

Table 13

Experiment Four Priority Schemes

	Scheme 1	Scheme 2	Scheme 3
Priority-	1. Master Exec.	1. Master Exec.	1. Master Exec.
	2. Sim. Exec. Con. Head Monitor	2. Sim. Exec.	2. Con. Head
		3. Monitor	3. Monitor Sim. Exec.
		4. Con. Head	

Table 13 (continued)

Scheme 1	Scheme 2	Scheme 3
3.Display	5.Display	4.Display

The prioritizations were implemented by assigning values to Attribute 2 at the source nodes for signals using the various executive programs. The queue for the CPU was then changed from FIFO to S/2, in order to allow the smallest values of Attribute 2 to move to the front of the queue.

These three schemes were tested for the original parametric model and for each maximum CPU run time determined in Experiment Three. The intent was to observe the effect of the prioritization schemes at the maximum CPU run time boundary levels. Graphical results of this experiment are depicted in Figure 85. Only the CPU run time with the most profound effect is illustrated. The complete set of graphs is located in Appendix E. The most noticeable effects of prioritization were observed when the priority schemes were tested for the system at the maximum allowable CPU run time for Altimeter and Clock signals. All three priority schemes greatly increased TIS well beyond the viability threshold for Monitor and PMIU signals and substantially above the viability threshold for Altimeter signals. The obvious effect of all priority schemes is a large increase in TIS for transactions. If priority schemes are, in fact, implemented in the actual Hot Bench, the maximum allowable CPU times will have to be adjusted downward to accomodate the situation. The system with the original parametric model still performs well under all three priority schemes.

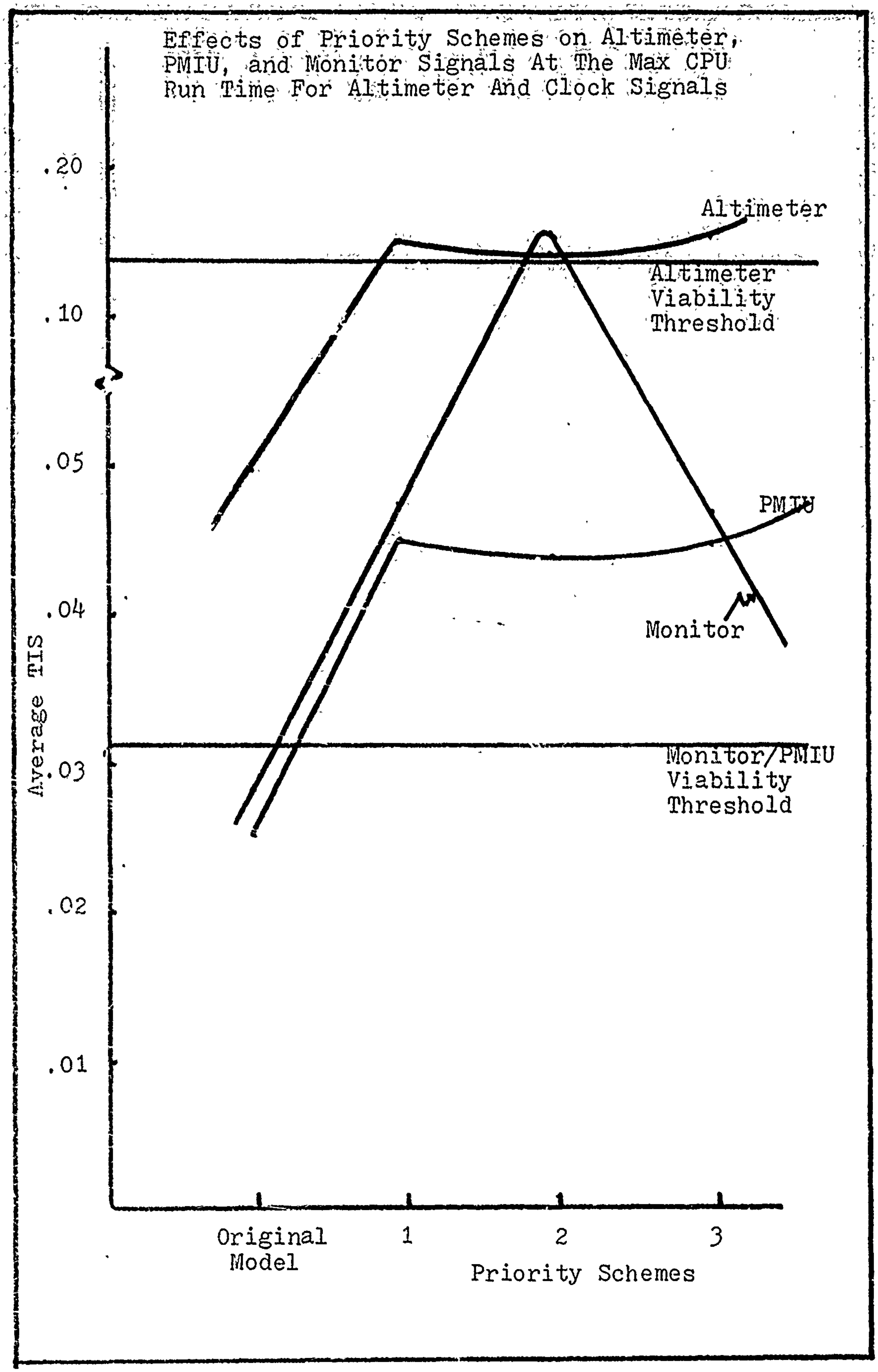


Figure 85. Results-Experiment Four

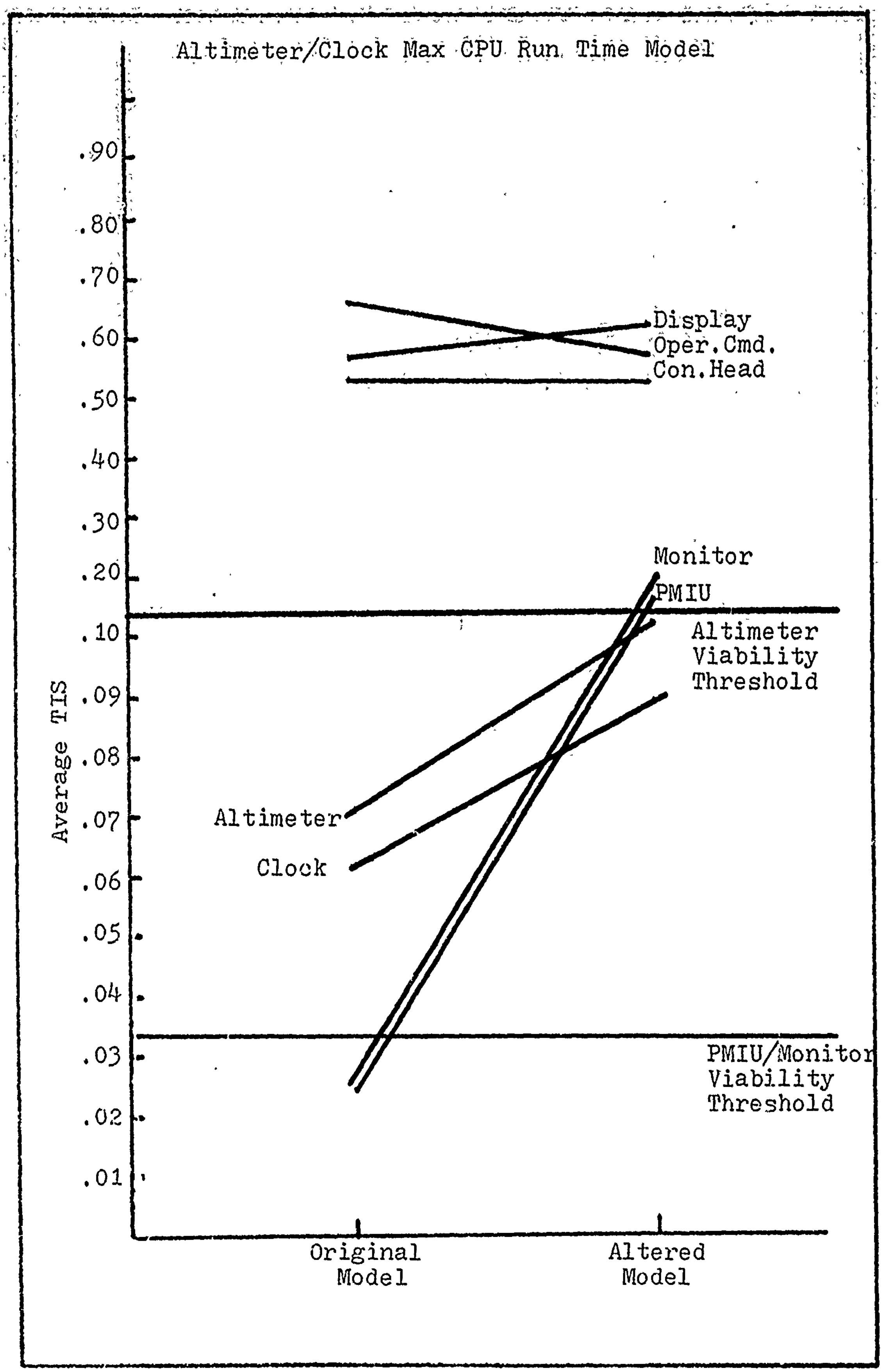


Figure 86. Results- Experiment Five

Of the three schemes tested, the first and third perform identically. The second scheme yields better results for some signals and worse for others, however, for the most sensitive signals (PMIU, Monitor, Altimeter), all three priority schemes perform similarly. Statistics from this experiment are included in Appendix E.

In Experiment Five an architectural alteration was accomplished in order to simulate adding more software to the Hot Bench. In the original model, Display, Altimeter, and Clock signals all require the Display Executive program in order to be processed. In Experiment Five, the addition of two more executive programs, called "Altimeter" and "Clock", is incorporated into the model. This required the addition of two more queues in which the signals would wait for the Executive Programs to be allocated, and two more allocate nodes for accomplishing the allocation. The network of free nodes was also altered to facilitate the freeing of the two new executive programs following CPU processing. Two new resources were added to represent the two new executive programs, and the parametric model was unchanged. These structural changes are shown in Appendix E.

The goal of Experiment Five was to determine whether the addition of new software would help or hinder system performance in terms of the viability criterion. The new model was run for the original parametric model and for each maximum CPU run time parametric model. The most radical effect upon the system is shown in Figure 86. Adding the two new programs proved detrimental to the system. The Monitor and PMIU total

TIS increased to a level five and a half times the viability threshold, when the maximum Altimeter and Clock CPU run time model was tested. Clearly, this is unsatisfactory performance. For the original parametric model there was no significant change in the TIS for any signal. Based upon these observations, the development of more executive software for the Hot Bench would be counterproductive.

Conclusions

Evaluation of these experiments yielded the following conclusions. The initial system proposal is viable, though the CPU processing times in the parametric model may be inaccurate. These service times can be increased substantially (up to 350 times in some instances) with system viability maintained.

Resources critical to system performance are the CPU and the five executive programs, however, contention for them does not become a key factor until CPU service times are increased. Critical factors are the, already established, interarrival rates and the CPU processing times. Flexibility in the CPU times has been proven. Signals which are most sensitive to changes in the critical factors are those with the lowest viability thresholds, the PMIU, Monitor, and Altimeter signals. According to the results of Experiments Four and Five, the most critical CPU service time is that for the processing of Altimeter and Clock signals. Interestingly, the original parameter value for this service time can be increased over 300 times with viability preserved. According to Experiment Three, the Monitor Signal's TIS was the most sensitive to all changes in CPU processing times.

The effect of priority schemes upon the system is varied for different signals, depending upon which CPU service time is altered. When the parametric model for the maximum Altimeter and Clock CPU service times was incorporated into the three prioritization schemes, the system became unviable for Monitor, PMIU, and Altimeter. The conclusion to be drawn from this experiment is that the net effect of any of the prioritization schemes is a slowing down of the TIS for these three signals. If a prioritization scheme is employed in the VAX Operating System, the CPU run time for the Altimeter/Clock signals would not be increasable to 300 times, though it may still be increased substantially, from the original parametric model.

The possibility of writing new executive programs for the Altimeter and Clock signals was eliminated from consideration by Experiment Five.

Summary

A predictive Q-GERT simulation model of SEAFAC's KC-135 Hot Bench was successfully constructed and run. Once the model was running properly, five experiments were conducted exploring the various functional characteristics of the Hot Bench Executive Software and alternative software architectures. Results showed that the system designed by SEAFAC will be viable though there are considerable time tolerances in the projected CPU run times of the software.

The successful completion of this Q-GERT model of the Hot

Bench Executive Software illustrates the direct application of the Q-GERT language to Hot Benching and AISF operations. Q-GERT simulations and models have applications in both the Analysis Phase and Design and Mechanization Phase of the AISF OFP change cycle. In the Analysis Phase Q-GERT may be used to accomplish Functional and Discrete Event Simulations. Q-GERT's statistics collection abilities enable it to be used effectively to construct the analytic models required by the Analysis Phase. The Q-GERT software can be hosted on the General Purpose Processor. This software analysis tool also has possible applications in the Design and Mechanization Phase. The ability to expand a basic Q-GERT model and to alter the parametric model for a system make it ideal for exploring different design strategies.

VII. AISFs Of The Future

Introduction

According to the TRW Long Range Logistics Support Plan For Embedded Computer Systems (Ref 20); recent government and industry studies indicate that the future promises a continuation of a rapidly evolving and expanding use of technology, an ever-increasing enemy threat with a corresponding demand for mission responsiveness, and continuing competition for scarce ECS support resources. Within the Air Force the current pressure for "better-faster-more with less" will, in all likelihood, continue. Characteristics of the foreseeable future ECS environment include: multi service/national use of ECS and ECS support systems, more complex weapon and support systems, and increased pressure for inter-operability and standardization of ECS and ECS support systems. (Ref 20:4)

In this chapter future support concepts will be examined. Some of the concepts rely on the building blocks proposed in Chapter III and some rely on presently undeveloped technology. To provide a background frame of reference for the new concepts, a brief discussion of today's support approach, with its shortcomings, will be presented. Then, two approaches for the future will be examined including modular and distributed AISFs. Finally, a short discussion of present standardization efforts in the avionics arena, and the impact of this standardization on AISFs, will be presented.

Today's ECS Support Approach

In the past AISFs were established to support a particular weapon system or a major subsystem. These point designs had many advantages leading to their ready acceptance. Relevent regulations tended to support an "umbrella" type management policy for specific support systems. The system-peculiar AISF gave the system manager and his engineering team complete control over the operation and capabilities of the AISF. This approach made staffing, budgeting, facility planning and other administrative activities relatively simple to accomplish. The system-peculiar support philosophy is best suited to responsive software support, in that all AISF capabilities and resources can be channeled into the support of a priority problem when and if necessary. (Ref 9:645)

Though the single system AISF is responsive it is also extremely expensive. Facilities, equipment, software, and manpower must be dedicated and, in some instances, duplicated for each new system. The development and lack of sharing of unique, highly capable, and responsive software tools only multiplies the support cost. The proliferation of the system-specific software tools in existence today could be slowed by the development of a system where common software tools are shared among the various AISFs. The development and sharing of common software tools will also facilitate the sharing of a variety of hardware support equipments including Data Base Management Systems, and Data Reduction and Analysis Systems. Standardization of AISFs will also increase manpower productivity. A common pool of expertise shared among systems is more cost-effective than establishing the expertise at each system-peculiar AISF. (Ref 9:645-646)

Future Support Approaches

Integrated Multi-System AISF. The Integrated Multi-System AISF proposed by Babiak and Pariott (Ref 9) is based upon a building block approach with the existing or planned AISF as a foundation. Many of the essential elements required are already contained in existing AISFs which will require little more than a change in the initial or dedicated AISF configuration to form the core of the Multi-System AISF.

The first building block, the Dynamic Simulation System (DSS) consists of three major components: the Simulation Host Processor (SHP), a hardware Interface Unit (IU), and the avionics systems themselves. The DSS is essentially an STB or an ITB. As in the ITB of Chapter III, the SHP is an off-the-shelf mini/midi computer providing the capability to drive the avionic computers in a real time, closed loop, simulation mode, and to collect data from the running mission computers. Software resident of the SHP includes programs to control and monitor the interface to the avionics computers, programs to process data sent to and from the avionics subsystems, simulations for closed loop operation with the OFP, and support software.

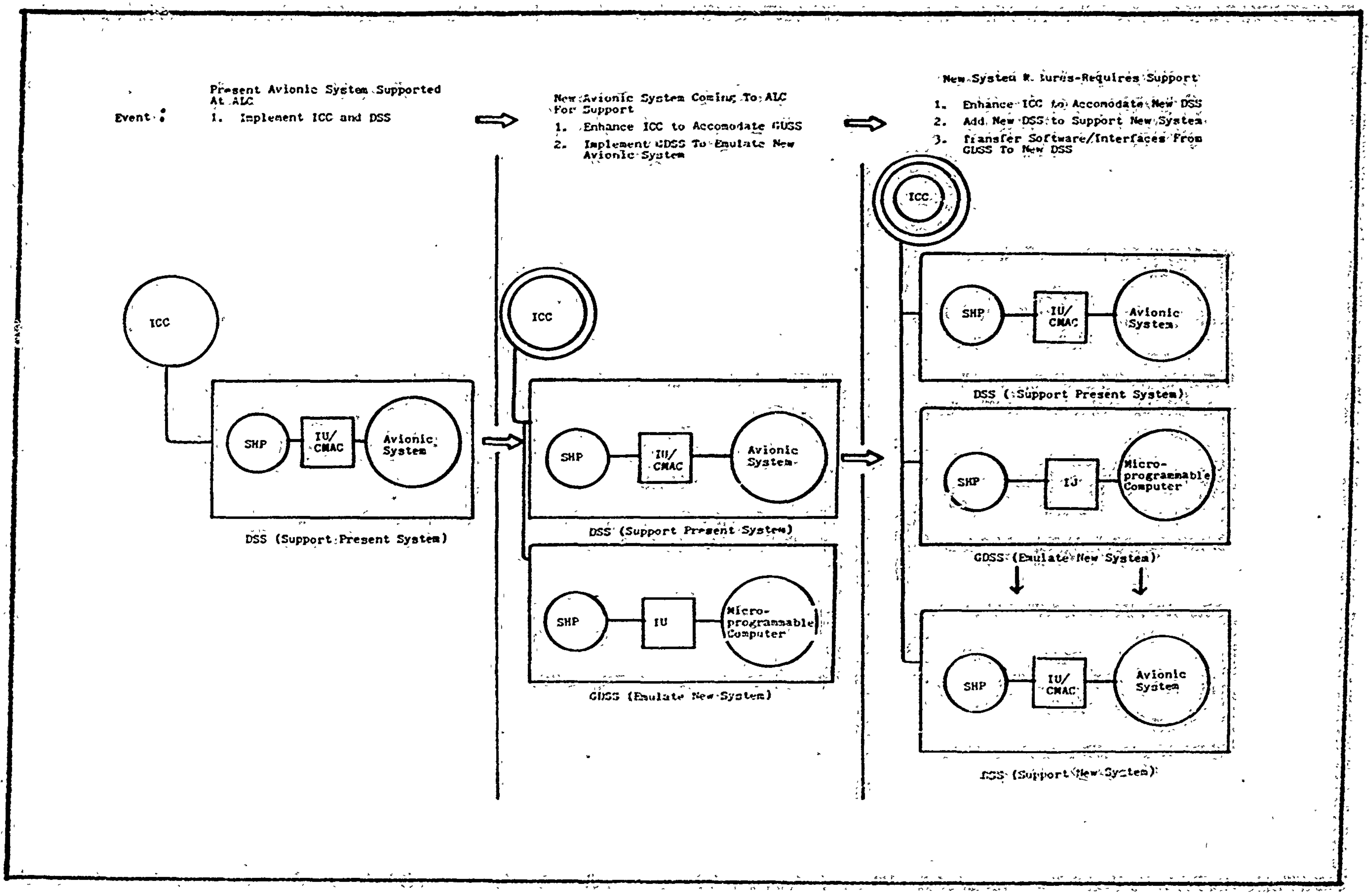
The hardware IU between the SHP and mission computers will implement loading, stopping, monitoring, controlling, stimulating, and displaying functions inherent to the simulation of the avionics suite. The interface also incorporates a CMAC facility for monitoring and controlling the mission computer's CPU and internal status.

The final components of the DSS are the avionics subsystems themselves. These may be composed of the actual hardware and/or scientific simulations.

The second building block in the Multi-System AISF is the Integrated Computational Complex (ICC) which provides the same capabilities as the General Purpose Computing Facility discussea in Chapter III. The ICC could contain a number of computers interfaced to the DSS for loading and unloading off-line batch processing and could be used primarily for the development of support software. The ICC could also be used for post run data reduction and analysis, documentation, data base management, and configuration management.

The DSS and ICC form the minimum configuration for the Integrated AISF and provide the capabilities of our present system-peculiar AISFs allowing the orderly building block approach for new systems.

As additional weapon systems are allocated to a particular Air Logistics Center (ALC) the integrated AISF must be responsive. Instead of creating a new AISF to support the new system as in the past, the existing Integrated AISF must provide for early understanding and analysis of the new avionic system/subsystem, verification and validation, and establishing requirements for AISF support of the new system. The functions can be accomplished by the third building block the Generalized Dynamic Simulation System (GDSS), featuring a microprogrammable computer. The first two elements of the GDSS, the SHP and IU, perform the same functions as in the DSS. The third core element of the GDSS, the microprogrammable computer, provides for emulation of the avionic



Event:

Present Avionic System Supported At ALC

1. Implement ICC and DSS

New Avionic System Coming To ALC For Support

1. Enhance ICC to Accomodate GDSS
2. Implement GDSS To Emulate New Avionic System

New System Matures-Requires Support

1. Enhance ICC to Accomodate New DSS
2. Add New DSS to Support New System
3. Transfer Software/Interfaces From GDSS To New DSS

ICC
SHP
IU/CMAC
Avionic System
DSS (Support Present System)
ICC
SHP
IU/CMAC
Avionic System
DSS (Support Present System)
SHP
IU
Micro-programmable Computer
GDSS (Emulate New System)
ICC
SHP
IU/CMAC
Avionic System
DSS (Support Present System)
SHP
IU
Micro-programmable Computer
GDSS (Emulate New System)
SHP
IU/CMAC
Avionic System
DSS (Support New System)

Figure 87. Development Of An Integrated AISF

computer, diagnostic monitor, and emulation of support software host processors. The diagnostic emulation will permit testing of the OFP without the actual target computer. Monitoring the avionic computer emulation will provide the same diagnostic benefits as those derived from monitoring the actual avionic computer. The ICC will also have to be upgraded to accomodate the added load imposed by the GDSS. A common family of computers should be used in the ICC to facilitate modularity.

The AISF proposed will support a digital avionic system or subsystem, and prepare for the future support of those systems earmarked for a particular ALC. As the new system matures, it will require its own DSS which will utilize the initial capabilities developed in the GDSS. Many of the support programs, and interfaces developed on the GDSS can be directly transferred to the new DSS because of the emulation of the mission computer on the GDSS. With the addition of the new DSS the ICC will again have to be updated. Figure 87 is a timeline showing the event-driven implementationof an Integrated AISF.

The Multi-System AISF has potential for high cost savings for the following reasons. Its building blocks are modular and can be added/deleted as requirements dictate. This ICC approach will force standardization of software including environmental models, operating systems, DBMSs, data reduction and analysis software, and programming languages. (Ref 9:649)

<u>Distributed</u> <u>AISF.</u> The concept of a Distributed AISF has been explored by TRW (Ref 4). The Distributed AISF would

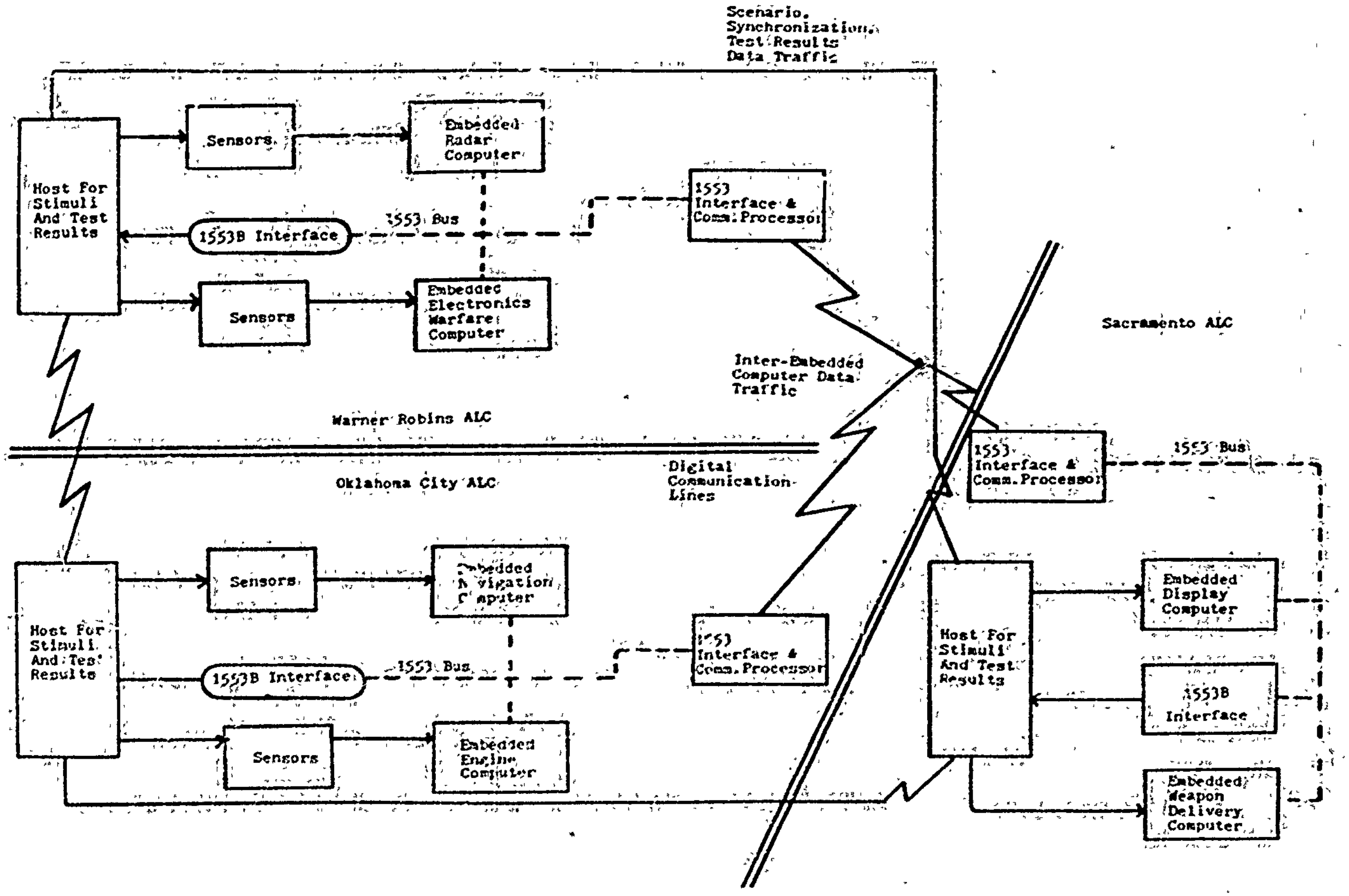


Figure 88. Hypothetical Example Of Distributed AISF (Ref 4:3-3)

be implemented using technology of the 1980-1990 period and would include embedded computers and host computers located at several ALCs interconnected by digital communication links. In this way the simulation of each item may located at the item manager's site and the simulation can be directed from a host computer at the site of the weapon system manager. (Ref 4:3-1)

In the following paragraphs a hypothetical example of a Distributed AISF is presented. The facility is diagrammed in Figure 88. Shown in the diagram are three ALCs, each supporting its respective subsystems. Installed at Warner Robins ALC are two embedded computers (Radar, Electronic Warfare) with associated live and simulated sensors. The SHP stimulates the sensors in accordance with mission scenarios and collects test results. The avionic components at the ALC are interconnected with a MIL-STD-1553 Multiplex Bus which carries outputs from the embedded computers to displays, weapons, and other computers on the bus.

Located at Oklahoma ALC are the embedded navigation computer and embedded engine computer with associated sensors and interfaced via the MIL-STD-1553 Multiplex Data Bus. A host computer stimulates the sensors and collects test results.

The embedded display computer and embedded weapon delivery computer are located at Sacramento ALC, (the site of the weapon system manager). The embedded computers are interfaced via the MIL-STD-1553 Multiplex Data Bus and receive stimulationfrom the host computer which directs the entire simulation (All three ALCs), including the remote test beds. All three ALCs are interconnected with digital communication links and the only data transmitted between centers are the data normally flowing on the aircraft data bus from one computer cluster to another, and the interhost synchronization data required to synchronize the mission scenarios at each ALC. Simulation results are stored in each host processor and after each run the raw or partly reduced data are transmitted to the weapon system host for final analysis. (Ref 4:3-3, 3-4)

The Distributed AISF approach has some advantages and disadvantages. It will facilitate shared use of expensive simulation facilities containing mission computers, sensors, and simulation software. Shared usage will be even more valuable when the MIL-STD-1750 computers and J-73/Ada languages become standardized. There will be few changes required to convert an item simulation from one system to another and the same personnel and test equipment can support them all. The Distributed AISF would also be administratively convenient because the system manager would not be required to be an expert in the item technologies. The item managers would control the simulation of only their devices. (Ref 4:3-1)

The disadvantages of the Distributed AISF approach lie in the cost and numerical availability of the communications system which would link the embedded and host computers. A complexity of scheduling between the item and system managers may also occur in scheduling and configuration of the systems.

Standardization

In the past, avionic system acquisitions have produced a continued proliferation of unique avionic systems and subsystems leading to ever-increased support and retrofit costs. These support costs can be significantly reduced for some subsystems by standardization. (Ref 4:8-1) There is an effort under way to achieve this standardization in the avionics as well as other military applications. Some current standardization efforts are discussed in the following paragraphs.

Standard Instruction Set Architecture (ISA). MIL-STD-1750 defines a standard 16-bit word Instruction Set Architecture (ISA) primarily for avionic applications. This standard defines the instruction set that any program written in the standard computer language will execute or any computer conforming to the standard with some minor changes, provided the input-output subroutines (usually written in assembly language) are isolated from the OFPs, will execute. An update to MIL-STD -1750, MIL-STD-1750A, was introduced following the discovery of deficiencies in MIL-STD-1750 during the testing of various implementations of the ISA. An extensive set of software tools is now being developed for MIL-STD-1750A and a users group has been formed. Full test and certification of the standard is scheduled to occur in late 1981 at SEAFAC. (Ref 4:8-2)

Standard Data Bus. The MIL-STD-1553 Multiplex Data Bus has been developed to interconnect all airborne avionic components including stand-alone embedded computers. An application of the Data Bus was presented in Chapter V. The majority of avionic components will attach to the data bus no matter what their function or design. The advent of the standard data bus allows AISFs to use one bus to support many subsystems because driver logic and hardware are common. All AISFs which test arrays of computers can use the Data Bus or a simulated Data Bus to connect the devices at a single ALC. Finally, a standard interface could be defined from embedded microcomputers to the 1553B Chip, further standardizing the embedded microcomputer and simplifying the logistics of Bus Driver Chips. (Ref 4:8-3)

Standard Software Tools. The two technological trends in the AFLC support software include: standardizing high order languages (HOLs) and instruction sets permitting the same software tools to be used throughout the AFLC, and, long-range development of integrated tool sets (also called Integrated Software Support Environments). The support software tools being designed for the MIL-STD-1750A ISA include:

MIL-STD-1750A Simulator
MIL-STD-1750A Assembler/Cross Assembler
J-73 Compiler with MIL-STD-1750A Code Generator
Linker/Loaders
MIL-STD-1750A Acceptance Test Program

Standardization of languages and computer instruction sets will make possible the use of the same tools at all the AISFs. Tools can be readily transferred between facilities through an intercenter network. (Ref 4:8-5)

Other Standards. The proliferation of HOLs has been reduced through the development of the J-73 dialect of JOVIAL, a structured HOL derived form ALGOL. The J-73 dialect will be used to develop OFPs and suppport software for the next several years being replaced,at some point, by Ada. The development and use of a standard HOL will yield cost savings, reliability, and portability.

Ada was designed for reliability and maintainability and emphasizes program readability over ease of writing programs. For this reason Ada programs may take longer to write, but once written, are more likely to be correct and are easier to change. Ada incorporates useful forms of redundancy and its compilers must check for consistency. Ada utilizes English-like constructs instead of obscure abbreviations, and notations which have led to errors in past HOLs have been eliminated. Ada supports economical program development and maintenance by allowing programs to be made up of many separately-compiled parts without sacrificing its consistency checking properties.

Ada has received strong support from the Department of Defense and is rapidly being accepted by industry. These factors indicate that it is likely to succeed as a common language for both general-purpose and military applications. The U.S. Army is currently in the process of developing an Ada compiler for the VAX 11/780.

A standardized language and programming environment must be developed that will allow Ada's advantages to be realized in practice. (Ref 4:6-3)

Development of a library of standard application software for AISF use would also prove beneficial. Presently, most AISFs develop their own vehicle models, weapon models, and environmental models required for testing on ITB/STBs. Each unique model must be developed, coded, tested, debugged, integrated, and maintained. As a standard HOL becomes used and skilled software manpower becomes more scarce, it will become important to develop a library of models to be shared by the AISFs. (Ref 4:8-6)

Summary

In this chapter a brief discussion of future concepts in avionic software support was presented. The chapter began with a cursory treatment of the AISF concept, past and present, and some of its pitfalls. With the present support concept in mind, two new approaches were presented for the coming decade, the Integrated Multi-System AISF, and the Distributed AISF.

The most attractive of the two alternatives appears to be the Integrated Multi-System AISF. The capability to "foresee" the support requirements of new avionics systems with its microprogrammable computers (GDSS) makes it essentially a standard AISF. Another factor strongly in favor of this approach, particularly in these budget-conscious times, is the fact that most of the equipment and software to implement the Integrated Multi-System AISF is already in place. Administratively and organizationally this approach would allow the current AISF-ALC relationship to continue. One advantage of the Distributed AISF not realized with the Integrated Multi-System approach is the fact that the system manager is relieved of the responsibility of being expert in all of the subsystem or item technologies. The fact that the Distributed AISF provides for the sharing of expensive computer and simulation facilities is also a benefit which may be outweighed, however, by the cost of developing and implementing the network communication system required. No matter which approach is applied to develop future facilities, the most important considerationis to develop the new facilities to support the avionic systems standards being implemented.

The final topic covered in this chapter was the standardization of avionic systems. This standardization is crucial to the realization of a standard AISF. All of the system-specific functional requirements discussed in Chapter III and modeled in Chapter IV were system-specific because there has been no standardization of systems in the past. As more avionic systems become standardized, these functional requirements will become universal and the standard AISF will become a reality.

VIII. Conclusions And Recommendations

Conclusions

All desired objectives were accomplished. The first objective, development of a complete design for an AISF and a complete set of functional requirements was accomplished in Chapter III. DFDs were chosen as the vehicle for presenting the functional requirements because of their structure and the ease with which they can be developed into an SADT model. The structure of the DFD approach allowed the functional requirements to be iterated from a high level to a detailed level. The DFD model revealed processes (functions) and data flows. The high-level functional requirements have universal application for all systems, however, as the functional requirements became more specific, the avionic system/subsystem being supported began affecting their implementation. For example, the characteristics of the avionics hardware and software employed by a specific system affect the choice of diagnostic capabilities for the ITB/STB. Characteristics of CMACs are presently determined by the avionic system/subsystem being monitored. Hardware interface unit characteristics and capabilities are also affected by the type of avionics system being supported. Finally, software, such as the number and characteristics of the ITB application software programs, is affected by the individual system/subsystem being supported. The proliferation of new systems has resulted in a proliferation of new AISFs.

The DFD functional requirements specification coupled with the Generic Resources List provided in Chapter III led directly to an SADT model implementation of such a facility. This was the second objective of the investigation. The SADT Actigrams provided an excellent vehicle for modeling the AISF. Activities, each with inputs and outputs, were depicted. In addition to this information, controls affecting the activities, and, mechanisms by which the activities may be accomplished, are provided. Again, many of the AISF's characteristics were determined by the avionic system being supported.

The third objective was to implement the ITB portion of the model developed in Chapter IV, and to provide SEAFAC a predictive model of their hot bench. The Q-GERT modeling and analysis language was ideal for this purpose. The simulation of SEAFAC's hot bench was developed and run to predict the performance of the actual facility presently being implemented. Five different experiments were run on the simulation with varying architectures and parametric models. Recommendations based upon results of the experiments are presented in the Recommendations section of this chapter.

The present system-peculiar AISF philosophy is resulting in the proliferation of expensive facilities and a duplication of much effort Air Force wide. The advantage of individual AISF responsiveness to a peculiar system problem is far outweighed by the tremendous cost of establishing a new facility for each new avionic system developed. As the level of embedded computer technology rises, the extent of embedded computer use aboard military weapon systems is inevitable. As the number of different avionic systems increases and the cost of software production and maintenance increases, the proliferation of AISFs must be stemmed. Present efforts to standardize avionic systems such as MIL-STD-1750A, MIL-STD-1553B, and standard HOLs are the first steps in the right direction towards the development of a standard AISF.

Two approaches for standardizing AISFs are the Integrated Multi-System AISF, which incorporates present technology and microprogrammable computer technology, and the Distributed AISF which would rely in the development and implementation of an intercenter communication network throughout the AFLC. Either approach may be viable for establishing a standard AISF, however, the Integrated Multi-System approach could probably be realized first due to the fact that much of the technology exists and many of the hardware components are already in place at the ALCs. Utilizing this approach, would yield one AISF at each of the ALCs instead of a new AISF for each new avionic system. Whichever approach is applied to render a standard AISF, the facility developed must be dedicated to supporting avionic systems dictated by the MIL-STDs.

Recommendations

The SADT model developed during this investigation is applicable to all AISFs and should be used in their implementation. A recommendation for future research in this area is to develop the facility, hardware, and personnel aspects of the model more fully, to complement the software model provided here. As the standard AISF becomes more realizable the model should be revised to reflect its characteristics.

Recommendations concerning the Hot Bench, being implemented at SEAFAC, are focused on the areas that affect system performance the most. The Hot Bench is viable, and, should the original parametric model be proven inaccurate, there is a great tolerance to increases in CPU times available in the system as designed. Since the interarrival rates are specified as functional requirements, the system must be designed to perform for the specified interarrival rates. Given the original parametric model, the Hot Bench is viable ; however, the validity of this parametric model is questionable. The recommendation is made to conduct further experimentation to determine a more accurate parametric model. This will become an easier task as the Hot Bench software is designed and run on the VAX CPU, and accurate CPU service times can be recorded. As exact CPU service times are discovered, the parametric model can be updated and this model can continually be used to predict Hot Bench performance. Close attention should be paid to the CPU service time for the Clock and Altimeter signsls, since they appear to impact the TIS of the PMIU, Monitor, and Altimeter signals greatly.

The Q-GERT modeling and analysis language was a useful tool in the implementation of a portion of an AISF, the ITB. It may be useful in other aspects of AISF implementation as well. Q-GERT simply provides a means for simulating, on a computer, real-world situations and collecting statistical data. Its application in the AISF environment could extend through all phases of the AISF implementation process and subsequent operations performed there. Future research efforts should concentrate on the applicabi ity of Q-GERT to the entire AISF implementation process and on the OFP change process.

Continued support and promotion of hardware and software standards will lead to the development of standard environments reducing avionic system maintenance and support costs. With the standardization of avionic system components, such AISF components as CMACs and interfaces can also be standardized. Microprogrammable computers and emulations will also enable any type of avionics computer to be implemented at the AISF, even before the hardware itself has actually been developed. The effort to standardize the AISFs should continue. Adherence to avionics software and hardware standards should be assured and development of support facilities around these standards will result. The proliferation of system-specific AISFs will be stopped.

Bibliography

1. A Study of Embedded Computer Systems Support Volume I Executive Overview, TRW Defense and Space Systems Group Report, 34330-6003-UT-00, September 1980.

2. A Study of Embedded Computer Systems Support Volume IV Selected ECS Support Issues:Recommendations/Alternatives, TRW Defense and Space Systems Group Report, 34330-6003-UT-00, September 1980.

3. A Study of Embedded Computer Systems Support Volume VII Requirements Baseline: Operational Flight Programs, TRW Defense and Space Systems Group Report, 34330-6003-UT-00, September 1980.

4. A Study of Embedded Computer Systems Support Volume VIII ECS Technology Forecast, TRW Defense and Space Systems Group Report, 34330-6003-UT-00, September 1980.

5. Airborne Systems Software Acquisition Engineering Guidebook for Development And Support Facilities, TRW Defense and Space Systems Group Report ASD-TR-80-5027, September 1980.

6. Airborne Systems Software Acquisition Engineering Guidebook For Requirements Analysis and Specification, TRW Defense and Space Systems Group Report ASD-TR-79-5027, September 1978.

7. An Introduction to SADT Structured Analysis And Design Technique, SofTech Inc. Waltham, Mass., November 1976.

8. Avionics Integration Laboratory Functional Configuration and System Level Requirements, Simulation Technology Inc., Torrence, California, October 1979.

9. Babiak, N.J. and Parriott, L.D. Jr.,"An Integrated Multi-System Approach to the Support of Digital Avionics," Proceedings of the IEEE 1979 NAECON, Dayton, Ohio, 15-17 May 1979.

10. Bergland, Glenn D., and Gordon, Ronald D., Tutorial: Software Design Strategies, IEEE, New York, 1979.

11. Best, Maj., SON For Embedded Computer System Software Support, AFLC/XRX, Draft Memo, 15 May 1980.

12. Corder, David R.,"Oklahoma City Air Logistics Center Approach to Embedded Computer System Support," Proceedings of the IEEE 1980 NAECON, Dayton, Ohio, 20 May 1980.

13. Davis, Richard M., Thesis Projects in Science and Engineering A Complete Guide From Problem Solution to Final Presentation, St. Martins Press, New York, 1980.

14. Goodman S.E., and Hcdetniemi, S.T., Introduction to The Design and Analysis of Algorithms, Mc Graw Hill Book Company, New York, 1977.

15. Hobart, William C., Design of a Local Computer Network For the Air Force Institute of Technology Digital Engineering Laboratory, MS Thesis, Wright-Patterson AFB, Ohio: School of Engineering, AFIT, March 1981.

16. Howard, John E. Major USAF, "F-15 Software Support," Proceedings of the IEEE NAECON, Dayton, Ohio, 20-22 May, 1980.

17. Eliff, Richard J., "Joint Tactical Information Distribution System (JTIDS) Software Support Philosophy," Proceedings of the IEEE 1980 NAECON, Dayton, Ohio, 20-22 May 1980.

18. Integrated Support Software System (ISSS) For the AFWAL Avionics System Analysis and Integration Laboratory Request For Proposal, USAF Systems Command Aeronautical Systems Division/PMRLB, Wright-Patterson AFB, Ohio, August, 1980.

19. KC-135 Avionics Modernization Hot Bench System Specification, SEAFAC Specification 000-0000-00A, 23 February 1981.

20. Long Range Logistics Support Plan For Embedded Computer Systems (ECS), TRW Defense and Space Systems Group Report, May 1981.

21. Mc Donald, E.G. "Electronic Warfare Avionics Integration Support Facility," Proceedings of the IEEE 1980 NAECON, Dayton, Ohio, 20-22 May 1980.

22. Navy Airborne Software Support Activity, Software Management Branch Naval Missile Center, TP-73-60, Point Mugu, California, December 1973.

23. Patterson, A.E., "Air Force Integration Support Facilities; Their Total Utility," Proceedings of the IEEE 1980 NAECON, Dayton, Ohio, 20-22 May 1980.

24. Pave Tack Avionics Integration Support Facility Requirements Analysis Report on Life Cycle Requirements For OFP Support, TRW Report 30433-6001-TU-00, May 1977.

25. Predictive Software Cost Model Study Vol I Final Technical Report, Hughes Aircraft Company Report, AFWAL-TR-80-1056, June 1980.

26. Predictive Software Cost Model Study Vol II Software Package Detailed Data, Hughes Aircraft Company Report, AFWAL-TR-80-1056, June 1980.

27. Pritsker, A. Alan B., Modeling and Analysis Using Q-GERT Networks, Halsted Press, New York, 1979.

28. Rey, J.A."The Anatomy of an Avionics System Development and Integration Laboratory," Northrop Aircraft Division, Hawthorne, California, 1979.

29. Rey, J.A. and Thompson, R.K.,"Computer Monitor and Control A Flexible, Cost-Effective Implementation," Proceedings of the IEEE 1979 NAECON, Dayton, Ohio, 15-17 May 1979.

30. Software Acquistion Management Guidebook: Software Development and Maintenance Facilities, MITRE Corp. Report, ESD-TR-77-130, April,1977.

31. Statement of Work (SOW) Contractor Software Engineering Services, TRW Defense and Space Systems Group 26 Oct 1979.

32. Thornell, David V. "Avionics Computer System Support at Ogden ALC," Proceedings of the IEEE 1980 NAECON, Dayton, Ohio, 20-22 May 1980.

33. Vincen, Paul, M. "The Logistics of Software-AFLC in The 1980s," Proceedings of the IEEE 1980 NAECON, Dayton, Ohio, 20-22 May 1980.

34. Weinberg, Victor, Structured Analysis, Yourdon Press, New York, New York, June 1979.

Appendix A

Chapter III Functional Requirements Hierarchy

This appendix contains a listing of all of the functional requirements discussed and diagrammed in Chapter III using DFDs. The functions are presented in hierarchical order.

0. FSED Process
1. Determine System Requirements
1.1 Conduct System Analysis
1.2 Define System
1.3 Decompose Requirements
2. Determine Software Requirements
3. Determine Preliminary Design
3.1 Design OFP Algorithm
3.2 Plan For Testing
4. Determine Detailed Design
4.1 Design Program
4.2 Plan For Testing
5. Code and Debug
5.1 Code OFP
5.2 Test Modules
5.3 Generate Software
6. Integrate and Test
6.1 Integrate Software
6.2 Test and Analyze Results
6.3 Integrate System
7. Accept System
7.1 Validate System Using Hot Bench and Flight Tests
7.2 Analyze Test Results
7.3 Update Data Base
8. Requirements Verification
9. Design Verification
10. Program Verification
11. Validation

1. OFP Change Process
1.1 Analysis
1.1.1 Define Requirements
1.1.1.1 Analyze Change Feasibility
1.1.1.1.1 Examine Validity of Proposed Change
1.1.1.1.2 Consolidate Results
1.1.1.1.3 Examine Impacts of Changes
1.1.1.1.4 Determine Risks Involved in Change
1.1.1.1.5 Examine Scope of Changes
1.1.1.2 Define Change
1.2 Design and Mechanization
1.2.1 Preliminary Design
1.2.1.1 Design Algorithm
1.2.1.1.1 State Problem
1.2.1.1.2 Develop Mathematical Model
1.2.1.1.3 Design Effective Algorithm
1.2.1.2 Develop Algorithm Test Plan
1.2.1.2.1 Analyze and Determine Algorithm Complexity
1.2.1.2.2 Choose Test Inputs
1.2.1.2.3 Test For Performance Quality
1.2.1.2.4 Test For Computational Limitations
1.2.1.2.5 Produce Algorithm Test Plan
1.2.2 Detailed Design
1.2.2.1 Design Program
1.2.2.1.1 Define Logical Processes in Multi-Level DFDs

1.2.2.1.3 Pricesly Define Inputs/Outputs/Module Functions

1.2.2.1.4 Create Hierarchical, Modular System

1.2.2.1.5 Determine System Interfaces

1.2.2.1.6 Verify OFP Design

1.2.2.1.6.1 Perform Operating System Functions For SHP

1.2.2.1.6.2 Provide STB Diagnostics

1.2.2.1.6.2.1 Allow Interactive Control Between Operator and OFP/Avionic Computer

1.2.2.1.6.2.2 Select Data According to Event or Program Condition

1.2.2.1.6.2.3 Insert or Inject Data Into Avionic Computer

1.2.2.1.6.2.4 Select Data Type, Quantity, Time Period

1.2.2.1.6.2.5 Control Avionic Computer State

1.2.2.1.6.2.6 Examine Avionic Computer

1.2.2.1.6.2.7 Provide Snapshot of Cycles of Avionic Computer

1.2.2.1.6.3 Simulate Operational Controls and Displays

1.2.2.1.6.4 Simulate Environment

1.2.2.1.6.5 Perform Avionic Computer Functions

1.2.2.1.6.6 Provide SHP-Avionic Computer Interface

1.2.2.1.7 Create Detailed Specifications For Modules

1.2.2.1.8 Anticipate Serious Design Problem Areas

1.2.2.1.9 Determine User's Priorities

1.2.2.2 Develop Program Test Plan

1.2.2.2.1 Determine Test Strategy

1.2.2.2.2 Develop Modular Test Plan

1.2.2.2.3 Determine Test Plan

1.2.2.2.4 Determine Test Cases

1.2.2.2.5 Determine Test Case Inputs and Desired Outputs

1.3 Code and Unit Test

1.3.1 Code Modules

1.3.1.1 Code Modules

1.3.1.2 Compile Module Code

1.3.1.3 Correct Syntax Errors

1.3.2 Test Modules

1.3.2.1 Assemble Code

1.3.2.2 Link Modules

1.3.2.3 Implement Test Plan

1.3.2.4 Reduce and Analyze Data

1.3.2.4.1 Unpack Data Tapes

1.3.2.4.2 Edit and Format Data

1.3.2.4.3 Smooth Data

1.3.2.4.4 Perform Statistical Analysis

1.3.2.4.5 Format Data For Display

1.3.2.4.6 Plot and Print Display Quantities

1.3.2.4.7 Compare with Requirements

1.4 Testing and Verification

1.4.1 Integrate and Test ECS

1.4.1.1 Integrate Software and Test

1.4.1.1.1 Compile Modules

1.4.1.1.2 Assemble Modules

1.4.1.1.3 Link and Load Modules

1.4.1.1.4 Implement Test Plan

1.4.1.2 Analyze Program Test Results

1.4.1.2.1 Unpack Data Tapes

1.4.1.2.2 Edit and Format Data

1.4.1.2.3 Format Data For Display

1.4.1.2.4 Compare With Requirements

1.4.1.3 Integrate System and Test

1.4.1.3.1 Load OFP into Avionic Computer

1.4.1.3.2 Implement Integration Test Bed (ITB)

1.4.1.3.2.1 Implement Operational Controls and Displays

1.4.1.3.2.2 Perform SHP Duties

1.4.1.3.2.2.1 Perform Interactive Processing

1.4.1.3.2.2.1.1 Definc and Prepare Tests

1.4.1.3.2.2.1.2 Form Test Case Files

1.4.1.3.2.2.1.3 Manipulate Files

1.4.1.3.2.2.2 Perform Batch Processing

1.4.1.3.2.2.2.1 Facilitate Batch Job Submission

1.4.1.3.2.2.2.2 Develop Software

1.4.1.3.2.2.2.3 Perform Post Test Analysis

1.4.1.3.2.2.3 Perform Real-Time Processing Tasks

1.4.1.3.2.2.3.1.1.1 Support Real-Time Processing

1.4.1.3.2.2.3.1.1.2 Provide Peripheral Terminal Services

1.4.1.3.2.2.3.1.1.3 Support Interactive Processing

1.4.1.3.2.2.3.1.1.4 Support Batch Processing

1.4.1.3.2.2.3.1.1.5 Provide Program Generation Capability

1.4.1.3.2.2.3.1.1.6 Accomplish Files Management

1.4.1.3.2.2.3.1.2 Perform Executive and Control Functions

1.4.1.3.2.2.3.1.2.1 Control Program Ececution Sequence

1.4.1.3.2.2.3.1.2.2 Define Tests

1.4.1.3.2.2.3.1.2.3 Prepare

1.4.1.3.2.2.3.1.2.3.1 Test Equipment

1.4.1.3.2.2.3.1.2.3.2 Load Equipment

1.4.1.3.2.2.3.1.2.3.3 Equipment Initialization

1.4.1.3.2.2.3.1.2.3.4 Software Initialization

1.4.1.3.2.2.3.1.2.4 Select Executive Functions During Simulation Testing

1.4.1.3.2.2.3.1.2.5 Control Post Test Run Modules

1.4.1.3.2.2.3.1.2.6 Decode Mission Directive Data

1.4.1.3.2.2.3.1.2.7 Provide Data to Application Programs

1.4.1.3.2.2.3.1.2.8 Obtain Data For Post Test Analysis

1.4.1.3.2.2.3.1.2.9 Control Output of Simulation Data

1.4.1.3.2.2.3.1.2.10 Control Execution of Models

1.4.1.3.2.2.3.1.2.11 Process Data For Display

1.4.1.3.2.2.3.1.3 Perform Simulation Software Tasks

1.4.1.3.2.2.3.1.3.1 Provide Mission Profile Inputs

1.4.1.3.2.2.3.1.3.2 Simulate Aircraft Dynamics

1.4.1.3.2.2.3.1.2.3 Simulate Atmosphere and Weather

1.4.1.3.2.2.3.1.2.4 Simulate Avionic Sensor Outputs And Provide Inputs to Stimulate Sensors

1.4.1.3.2.2.3.1.3.5 Scale and Format Module Outputs

1.4.1.3.2.2.3.1.3.6 Define and Select Targets

1.4.1.3.2.2.3.1.3.7 Provide Error Generation and Real-Time Display

1.4.1.3.2.2.3.1.3.8 Generate Reference Data

1.4.1.3.2.2.3.1.3.9 Integrate Mission Profile Inputs And Transform to Postion, Velocity and Attitude

- 1.4.2 System Acceptance
  - 1.4.2.1 Validate System
    - 1.4.2.1.1 Test on ITB
    - 1.4.2.1.2 Flight Test
  - 1.4.2.2 Analyze Results
    - 1.4.2.2.1 Reduce and Analyze Data
  - 1.4.2.3 Update Data Base

Appendix B

SADT Tutorial

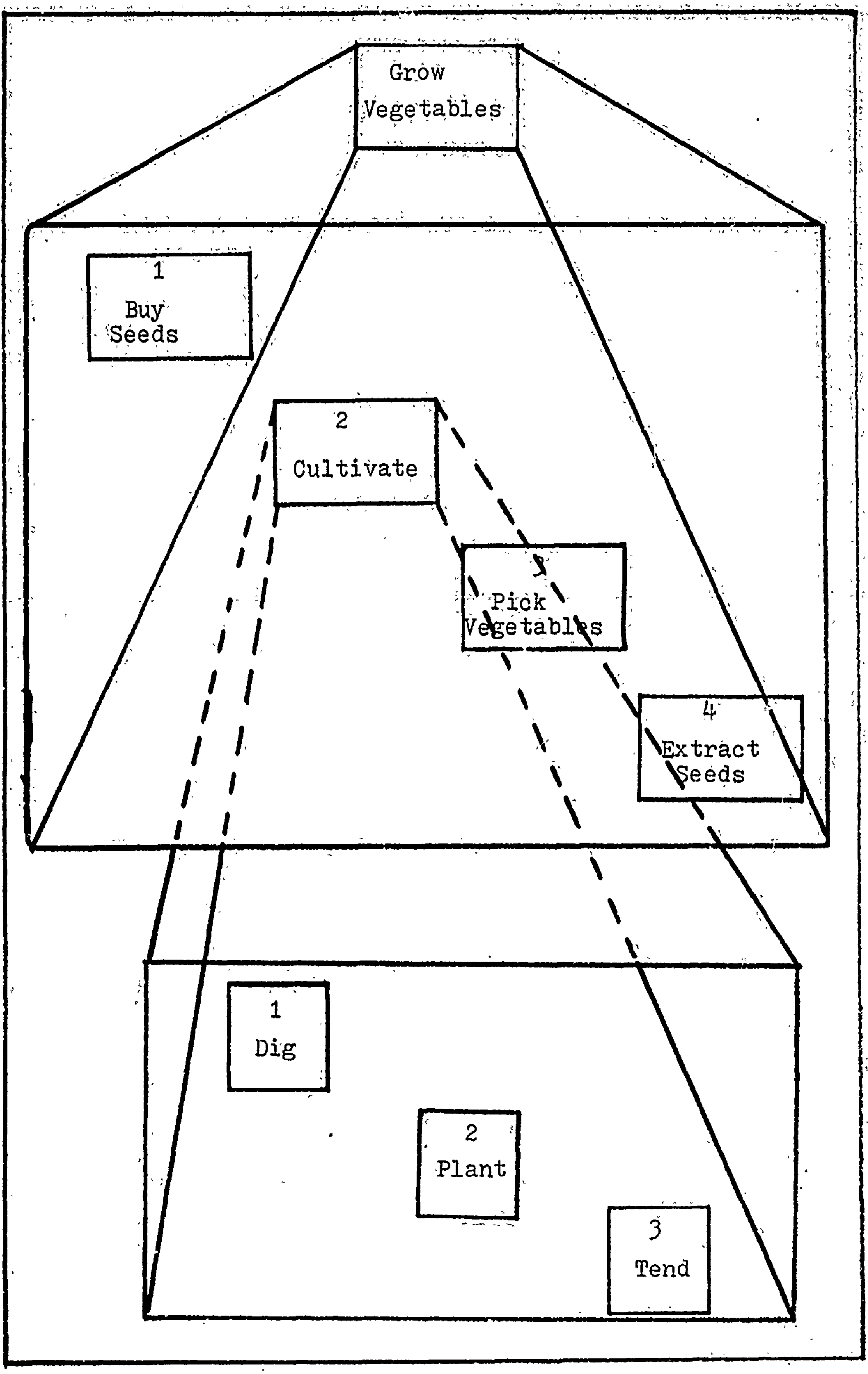
The Structured Analysis and Design Technique (SADT) is a comprehensive methodology for accomplishing functional analysis and systems design. SADT provides the techniques and methods for thinking in a structured way about complex problems and communicating analysis and design results in clear, precise notoation. (Ref 7:1-1)

A system may include any combination of hardware, software, and people. For new systems, SADT is used to analyze the requirements and functions and then to design an implementation which meets the requirements and performs the functions. For existing applications, SADT is used to analyze the purposes served by the application and functions it performs, and in addition, to record the mechanisms by which these are done. (Ref 7:2-2)

In modeling systems with the SADT subjects are decomposed into three to six major parts, then each of these parts are gradually decomposed, exposing more detail. The following figure shows the familiar subject "Grow Vegetables" decomposed into four parts. The second part "Cultivate" is in turn decomposed into three, more detailed, parts.

Reading Actigrams

An SADT diagram consists of boxes showing pieces of the decomposition and arrows connecting the boxes, which show how these pieces interface to one another. Descriptive labels



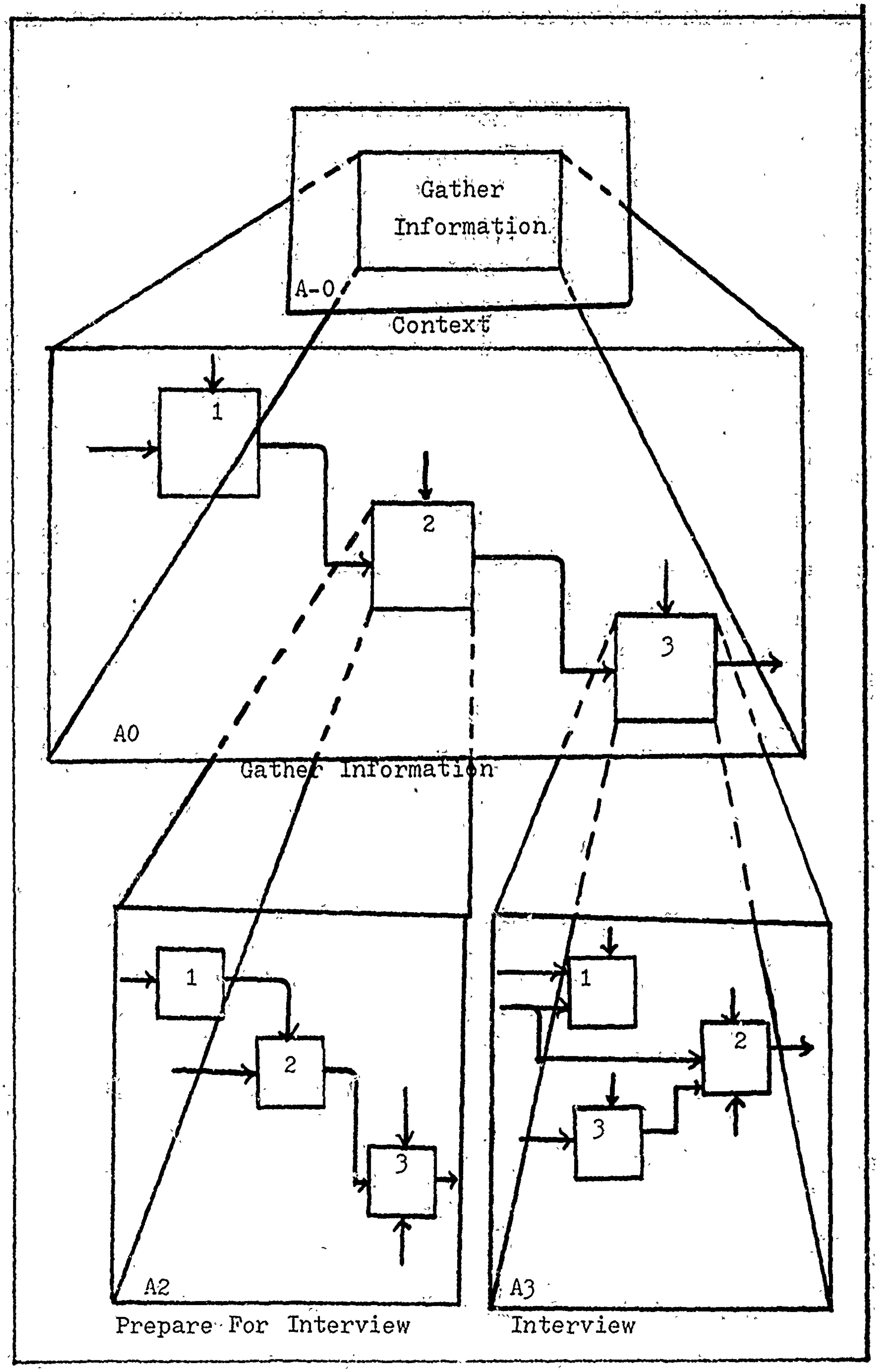
SADT Decomposition Process (Ref 7:4-1)

are written inside the boxes and along each arrow to describe its meaning. The side of the box at which the arrow enters or leaves, shows its interface role as an input, output, or control as shown in the following figure. Input data are transformed into output data by the activity represented by the box. Controls govern the way the transformation is accomplished and mechanisms provide the means for accomplishing the transformation. The arrows on an SADT diagram represent constraint relationships between boxes, not flow of control or sequence. Arrows entering a box show all that is needed by the box to perform its function. No box can do so until preceeding boxes have supplied needed results to it via arrows. Several boxes can fulfill their roles simultaneously, if the needed constraints have been satisfied. An output of one box may provide some or all to the inputs or controls required by one or more other boxes. (Ref 7:4-6)

Arrows represent classes of data, not individual data items flowing between boxes. Arrows are actually "pipelines" containing subclasses of data indicated by the arrow label. (Ref 7:4-7)

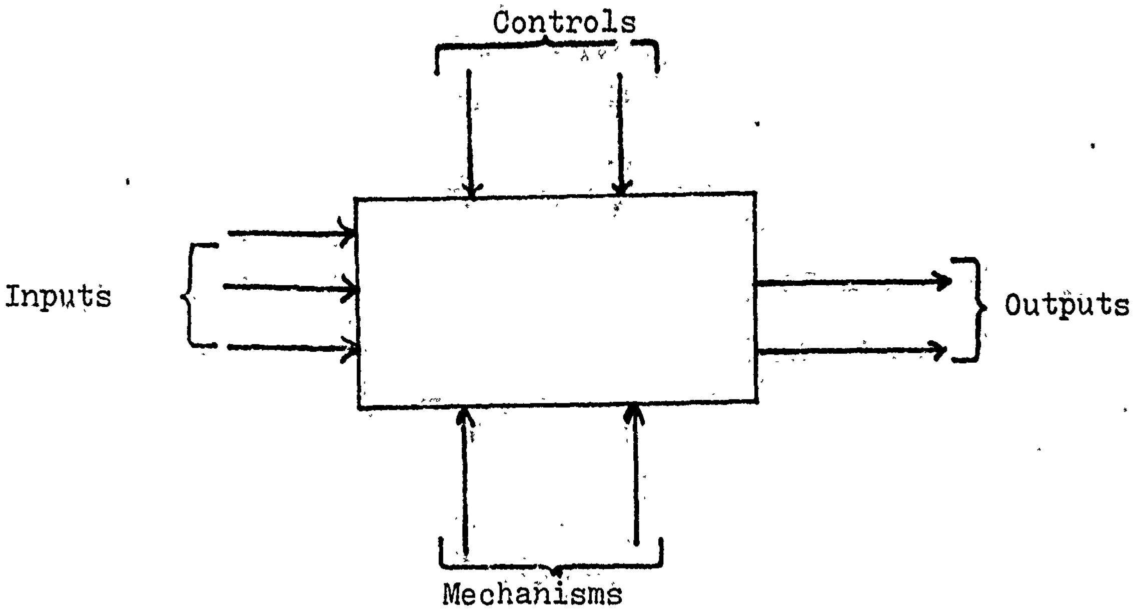
Diagram Numbering

The referencing system used in SADT is based on the hierarchy represented in the diagrams. The hierarchy is expressed by the assignment of a node number to each actigram. By convention, the top-level context actigram (a single box) has node A-0 (read A minus zero). Its child, the top level actigram, has node number A0, and its children are numbered



SADT Hicrarchy Showing Numbering Scheme

A1, A2, A3, etc.... The remaining boxes in a model are assigned node numbers as follows: Each box on a diagram contains a box number and each diagram has a node number in its lower left corner derived from its parent diagram and the specific box which it details. The diagram node number consists of the parent diagram's node number followed by the box number. This numbering system is shown in the figure on the following page.

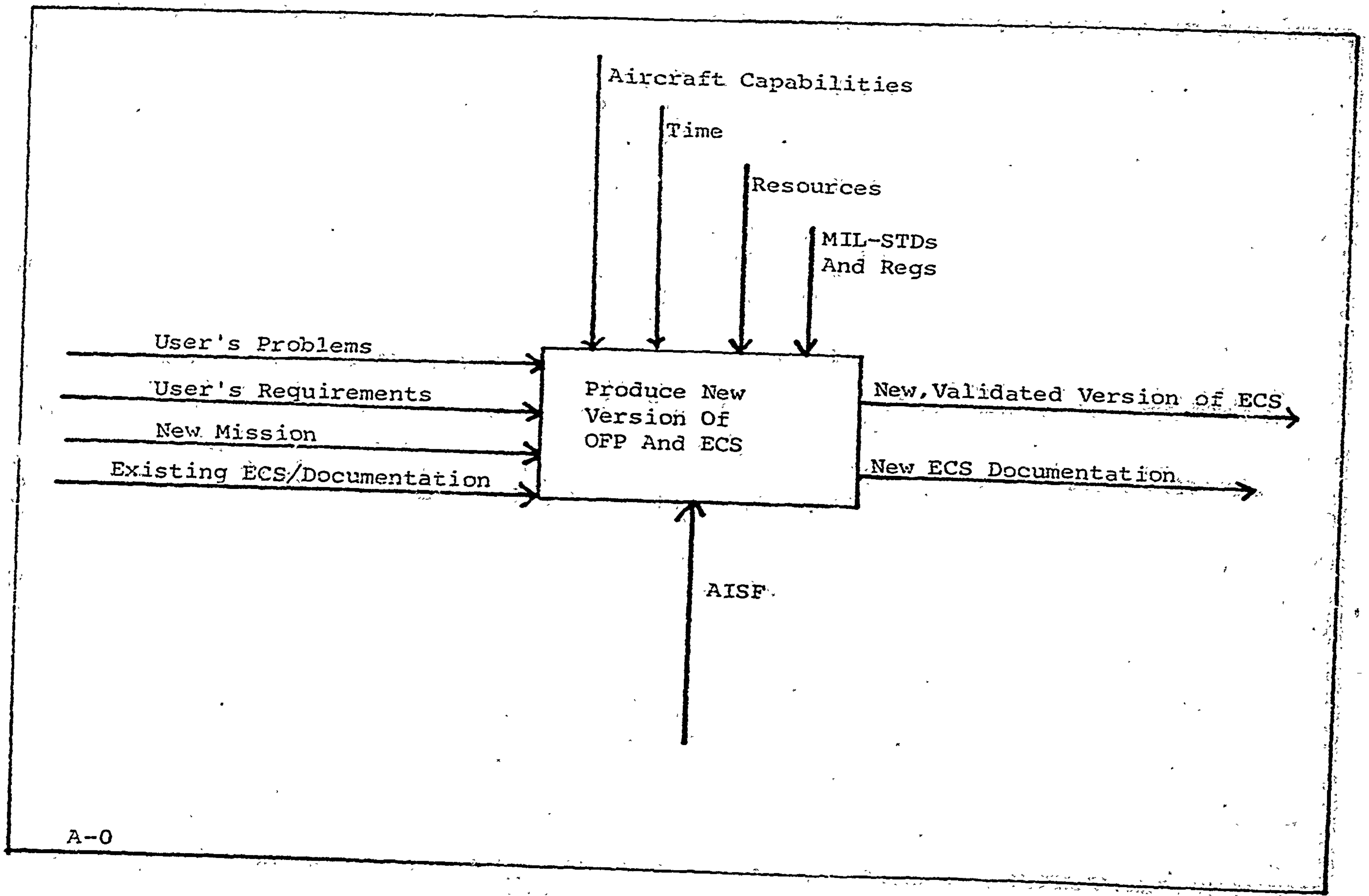


SADT Interface Arrows

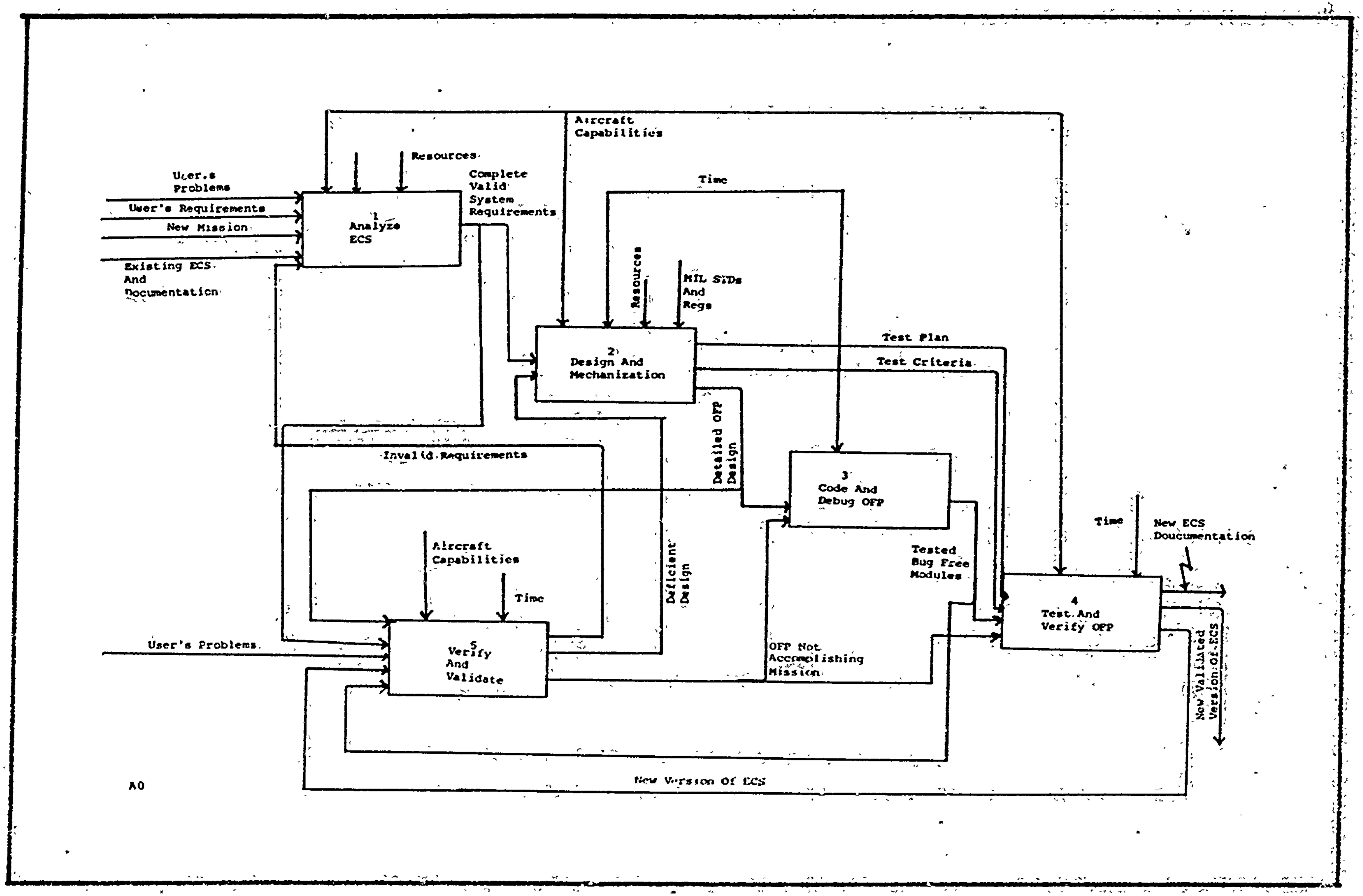
Appendix C

Complete SADT Actigram Model

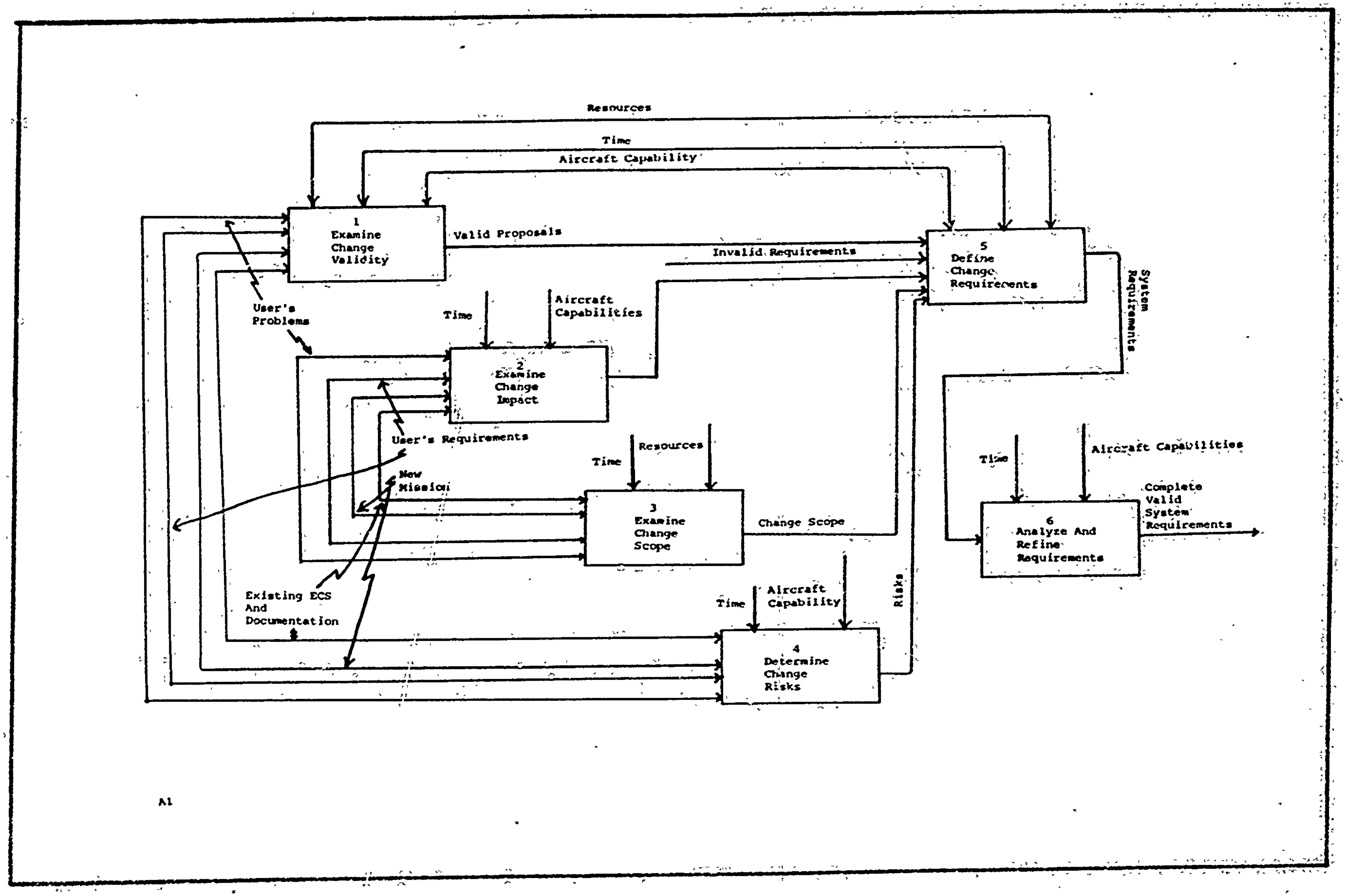
This appendix contains the complete SADT Actigram model of the functional requirements presented in Chapter III. SADT is a technique developed by the SofTech Corporation.



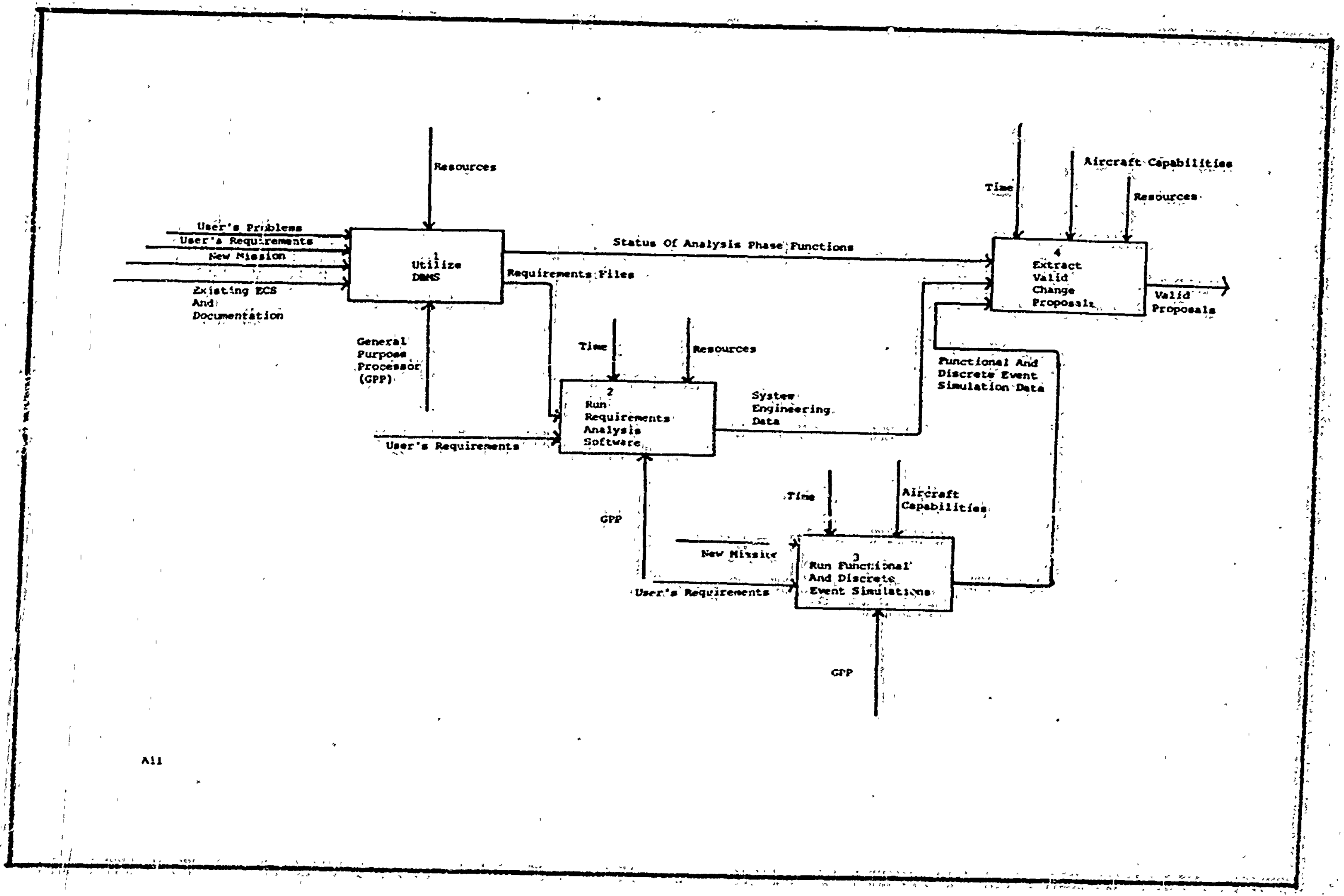
AISF Context



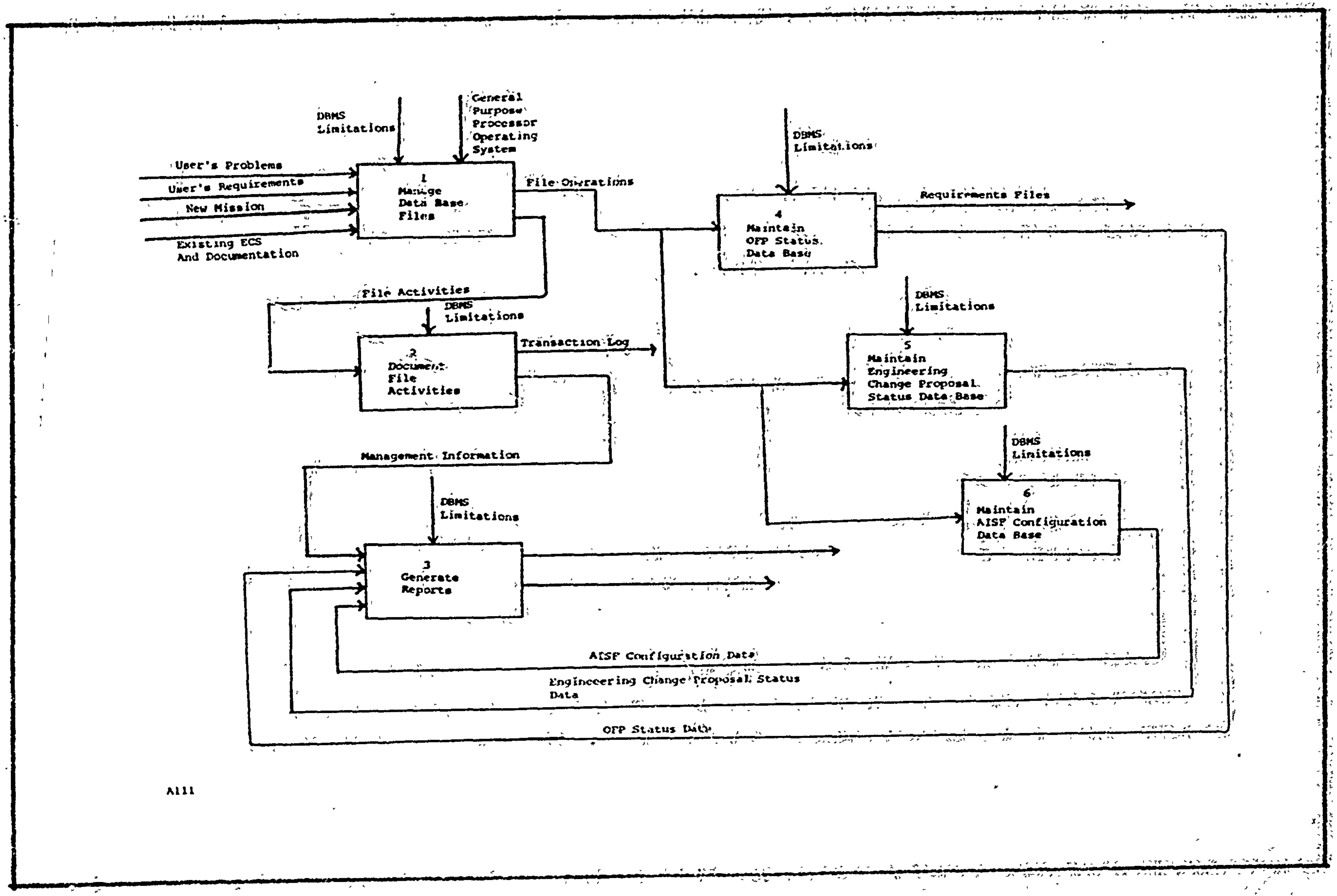
Major Functional Requirements For AISF


Resources
Time
Aircraft Capability
1
Examine
Change
Validity
Valid Proposals
Invalid Requirements
5
Define
Change
Requirements
System
Requirements
User's
Problems
Time
Aircraft
Capabilities
2
Examine
Change
Impact
User's Requirements
New
Mission
Time
Resources
3
Examine
Change
Scope
Change Scope
Time
Aircraft Capabilities
6
Analyze And
Refine
Requirements
Complete
Valid
System
Requirements
Existing ECS
And
Documentation
Time
Aircraft
Capability
Risks
4
Determine
Change
Risks
A1

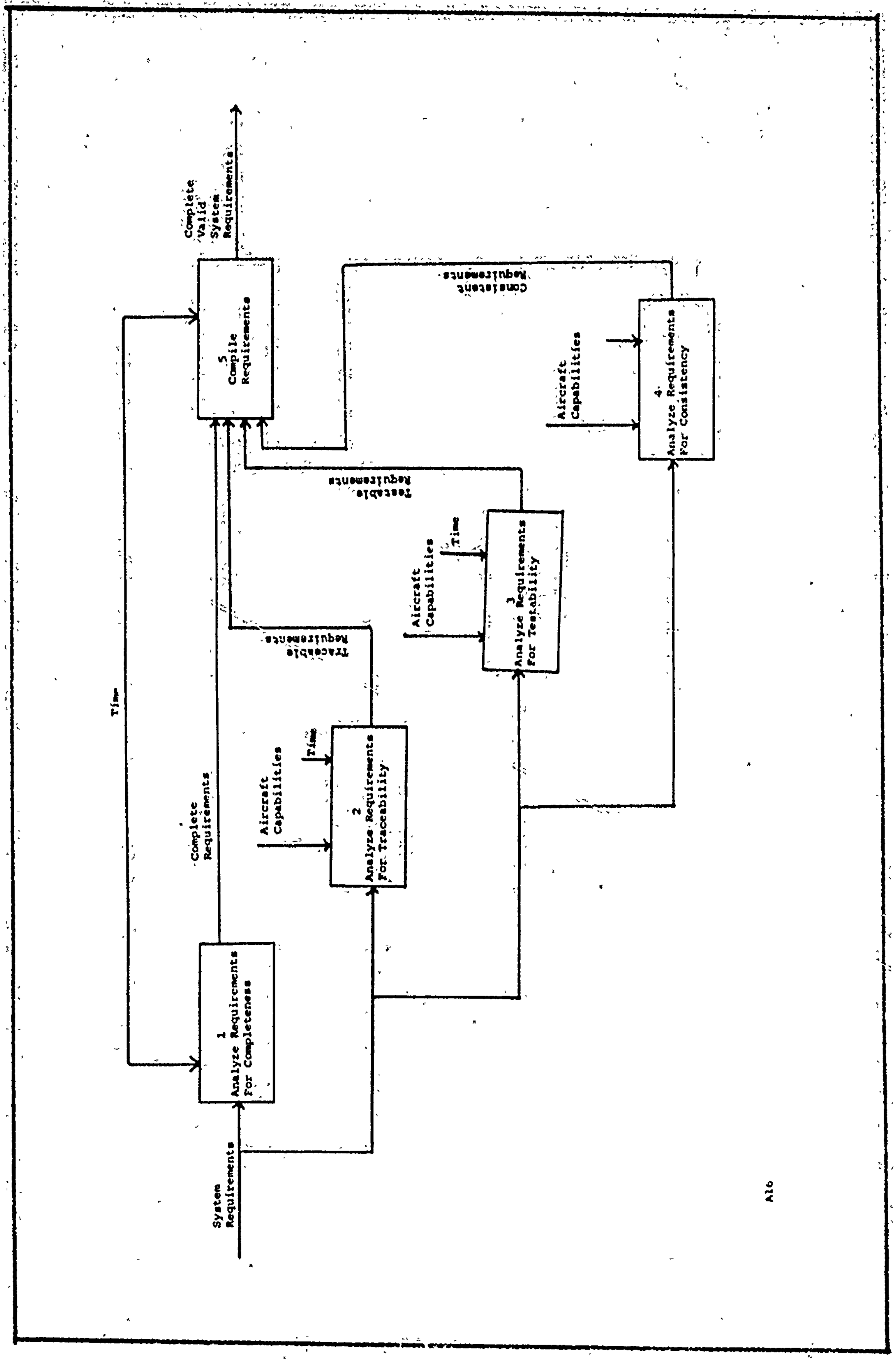
Analyze ECS


Resources
User's Problems
User's Requirements
New Mission
Existing ECS And Documentation
1 Utilize DBMS
General Purpose Processor (GPP)
Requirements Files
Status Of Analysis Phase Functions
Time
Resources
2 Run Requirements Analysis Software
User's Requirements
GPP
System Engineering Data
Time
Aircraft Capabilities
New Missic
User's Requirements
3 Run Functional And Discrete Event Simulations
GPP
Functional And Discrete Event Simulation Data
Time
Aircraft Capabilities
Resources
4 Extract Valid Change Proposals
Valid Proposals
A11

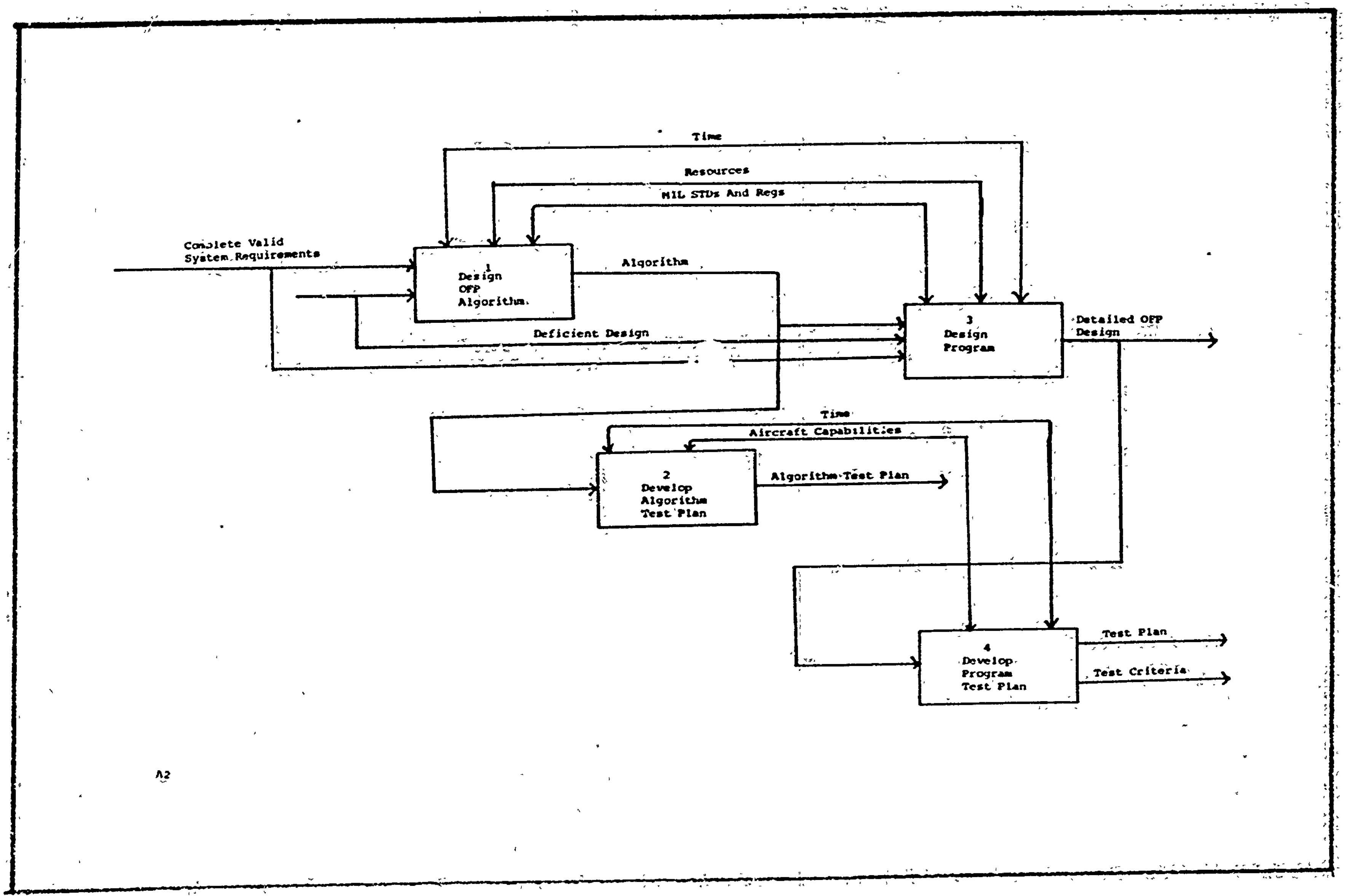
Examine Change Validity



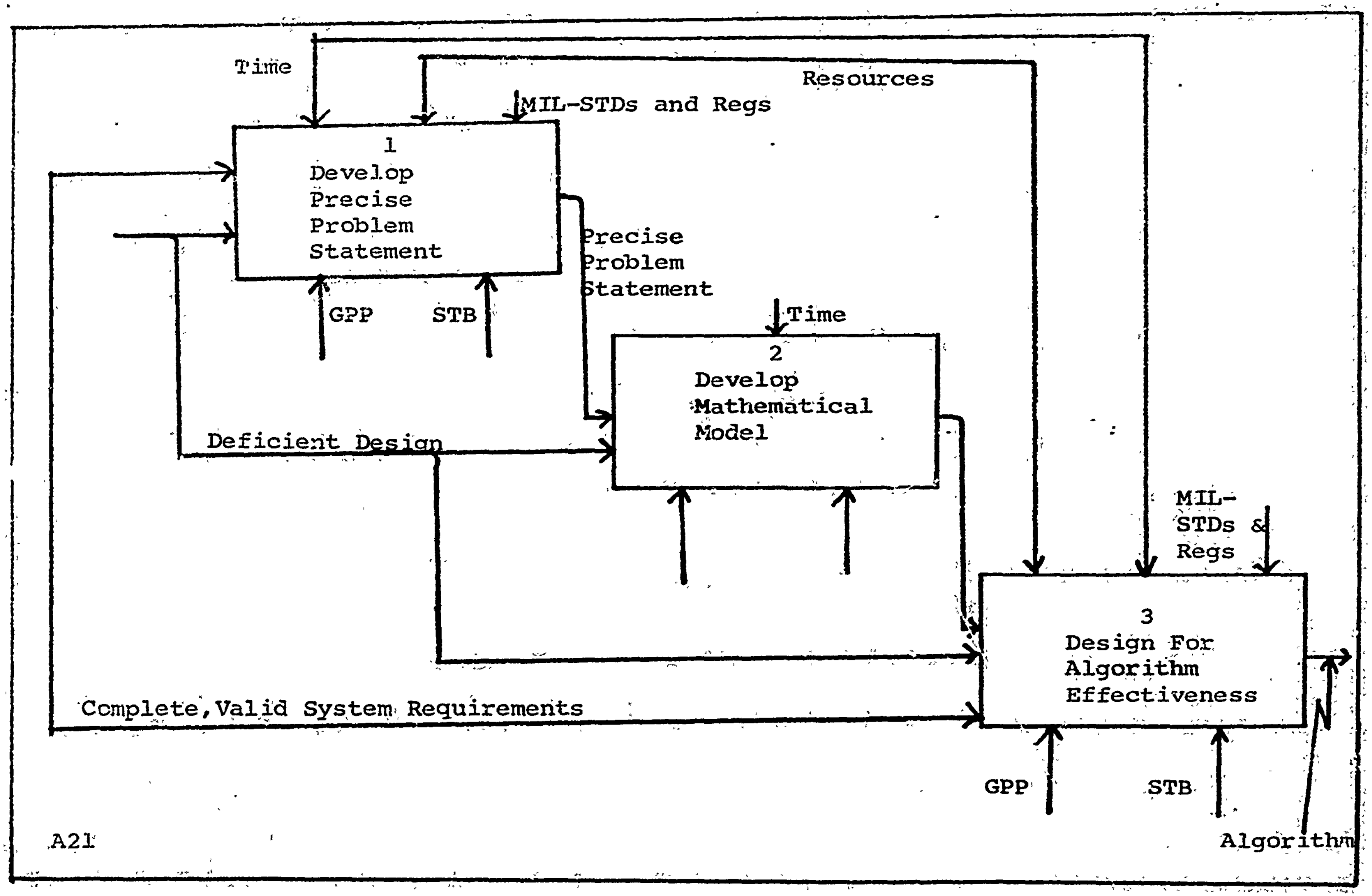
Utilize DBMS



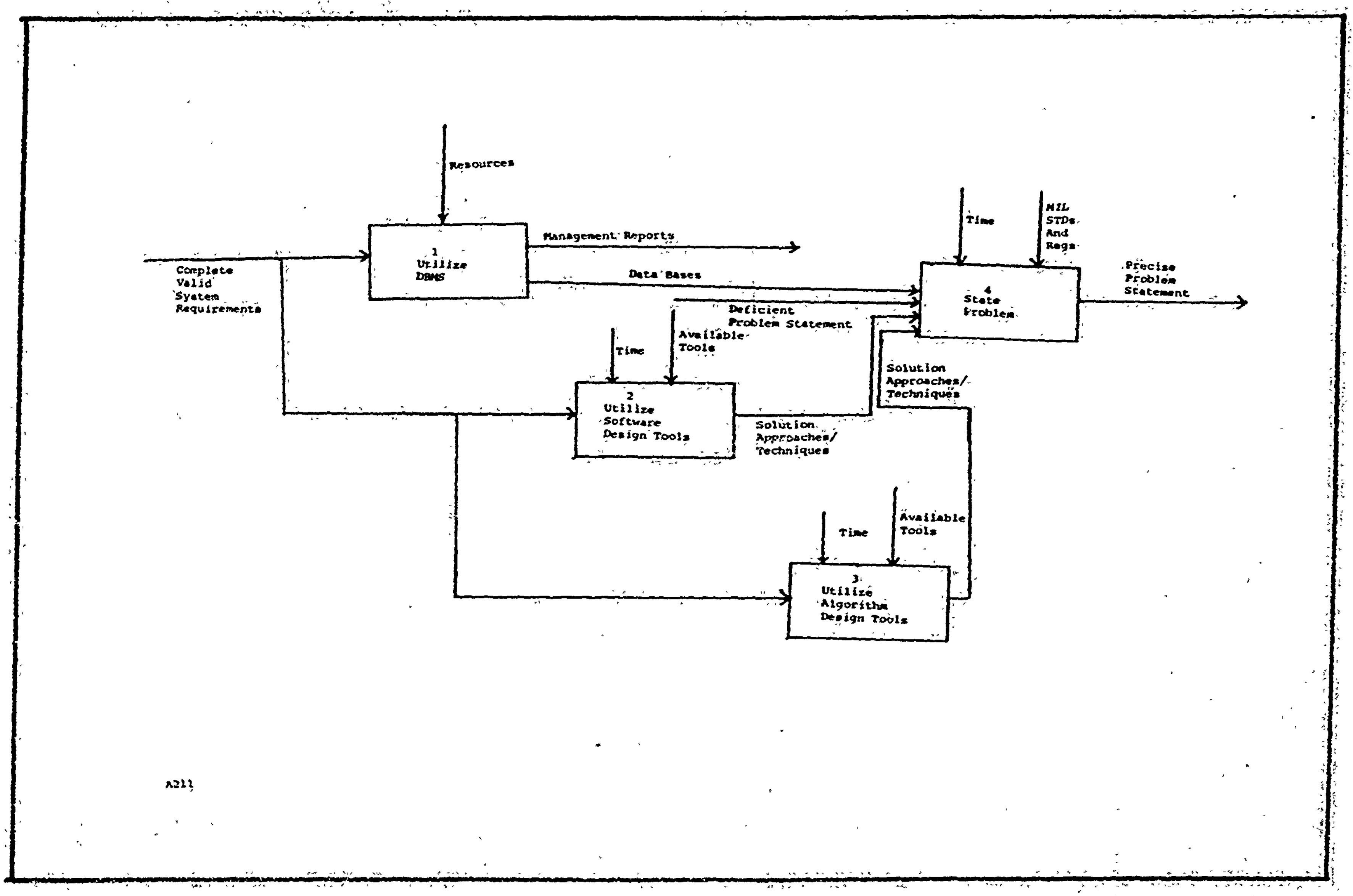
Analyze And Refine Requirements



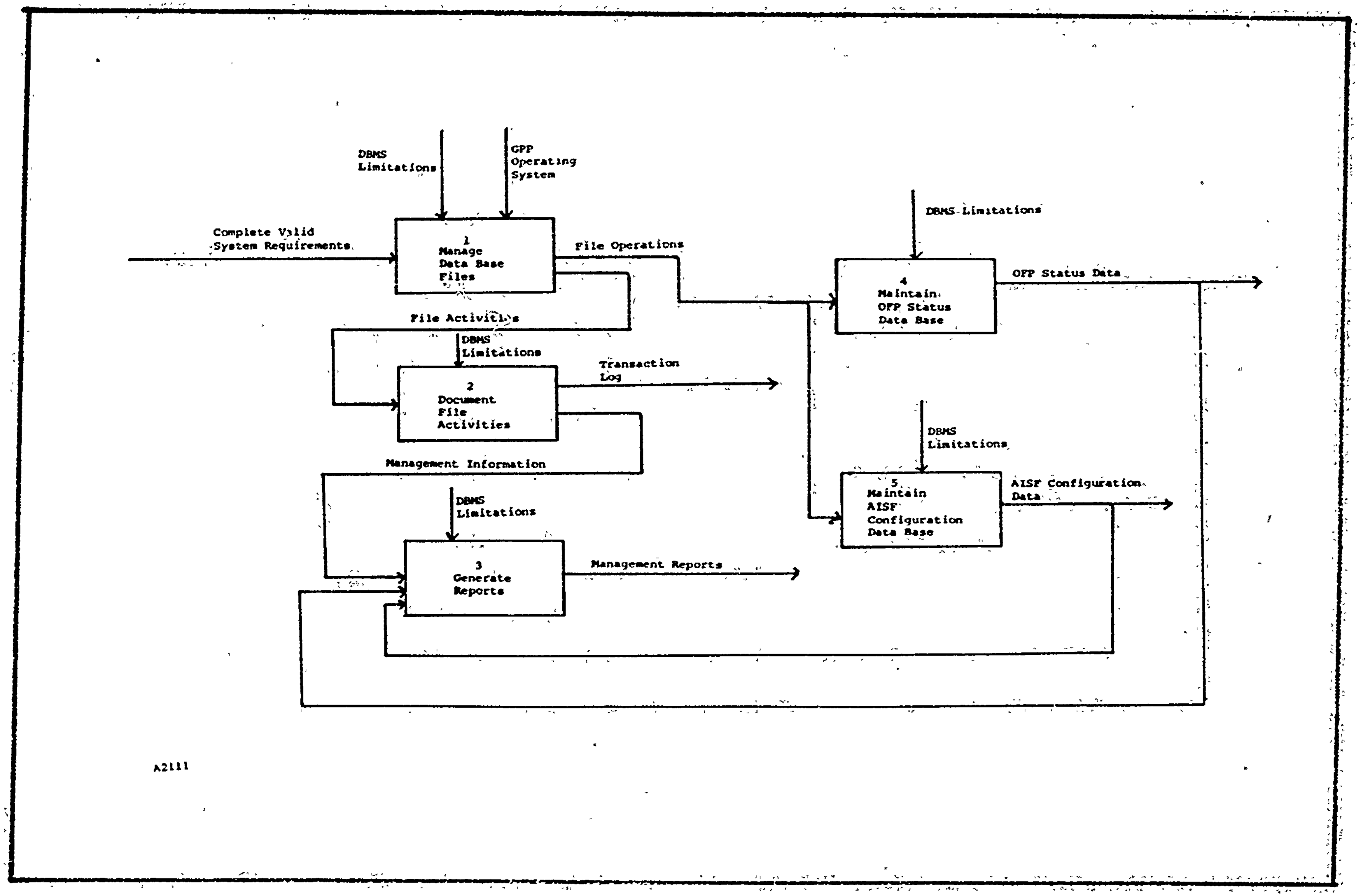
Design And Mechanization



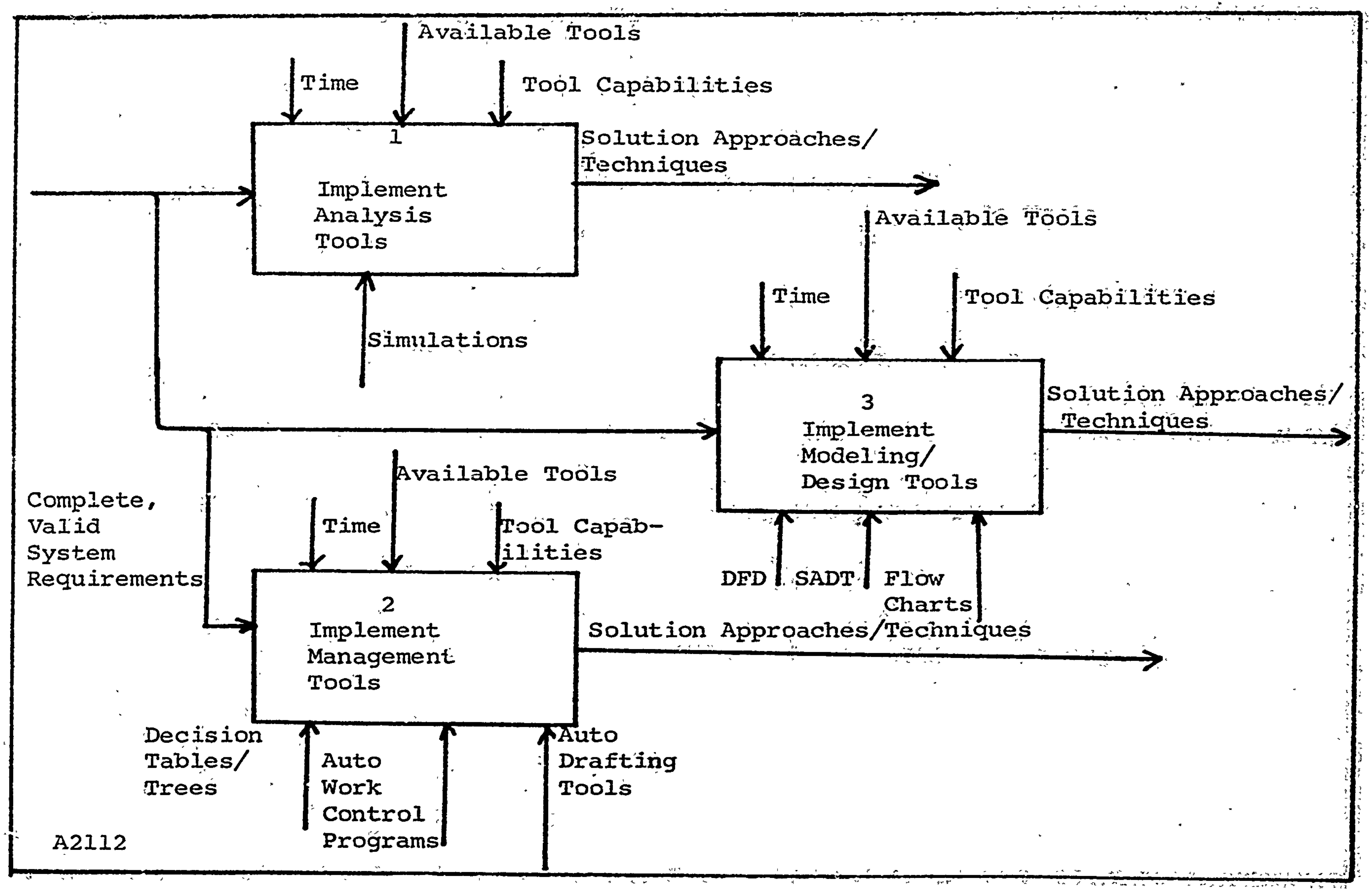
Design OFP Algorithm


Resources
1 Utilize DBMS
Management Reports
Data Bases
Complete Valid System Requirements
Time
MIL STDs And Regs
4 State Problem
Precise Problem Statement
Deficient Problem Statement
Time
Available Tools
2 Utilize Software Design Tools
Solution Approaches/ Techniques
Solution Approaches/ Techniques
Time
Available Tools
3 Utilize Algorithm Design Tools
A211

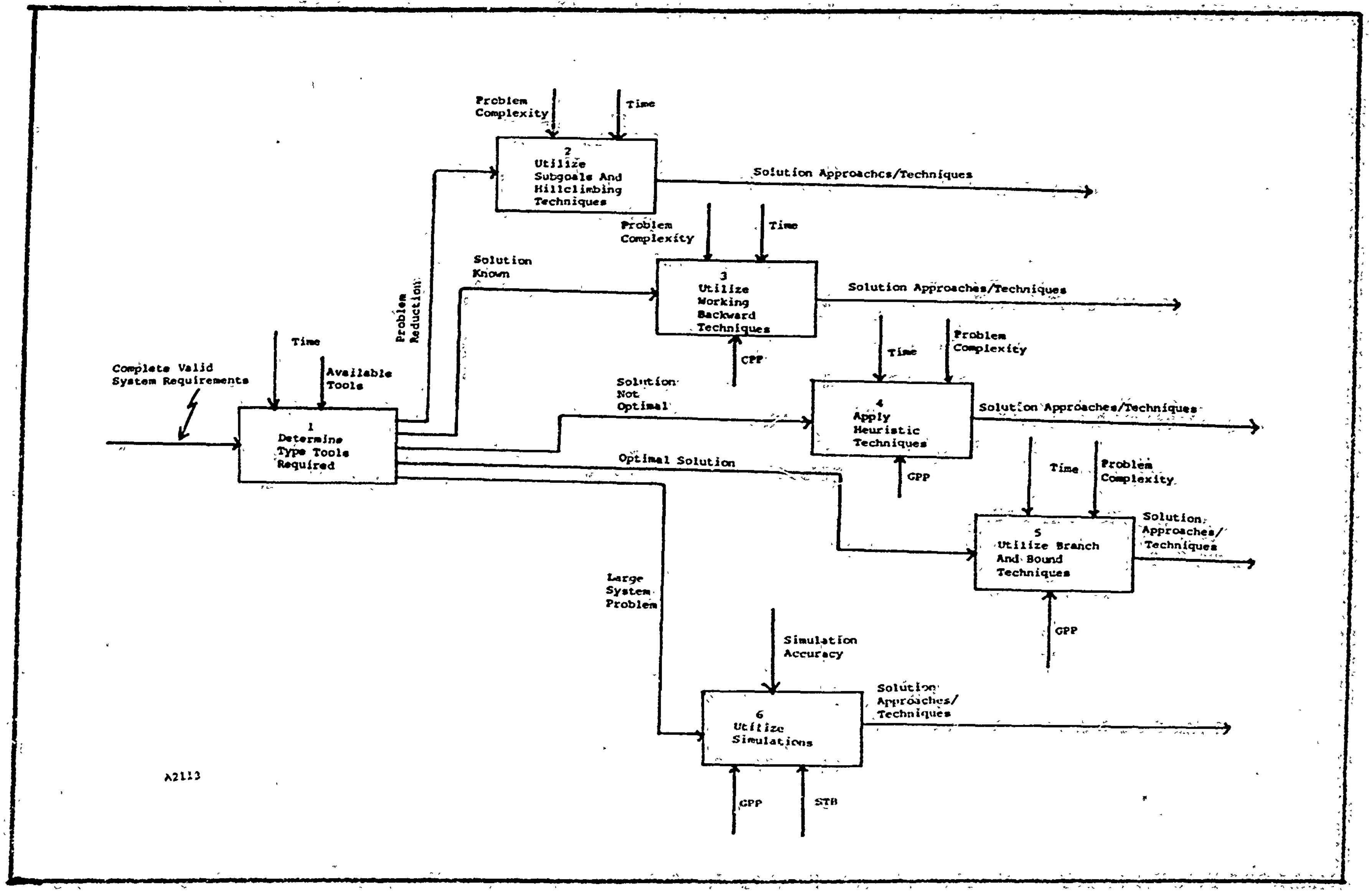
Develop Precise Problem Statement



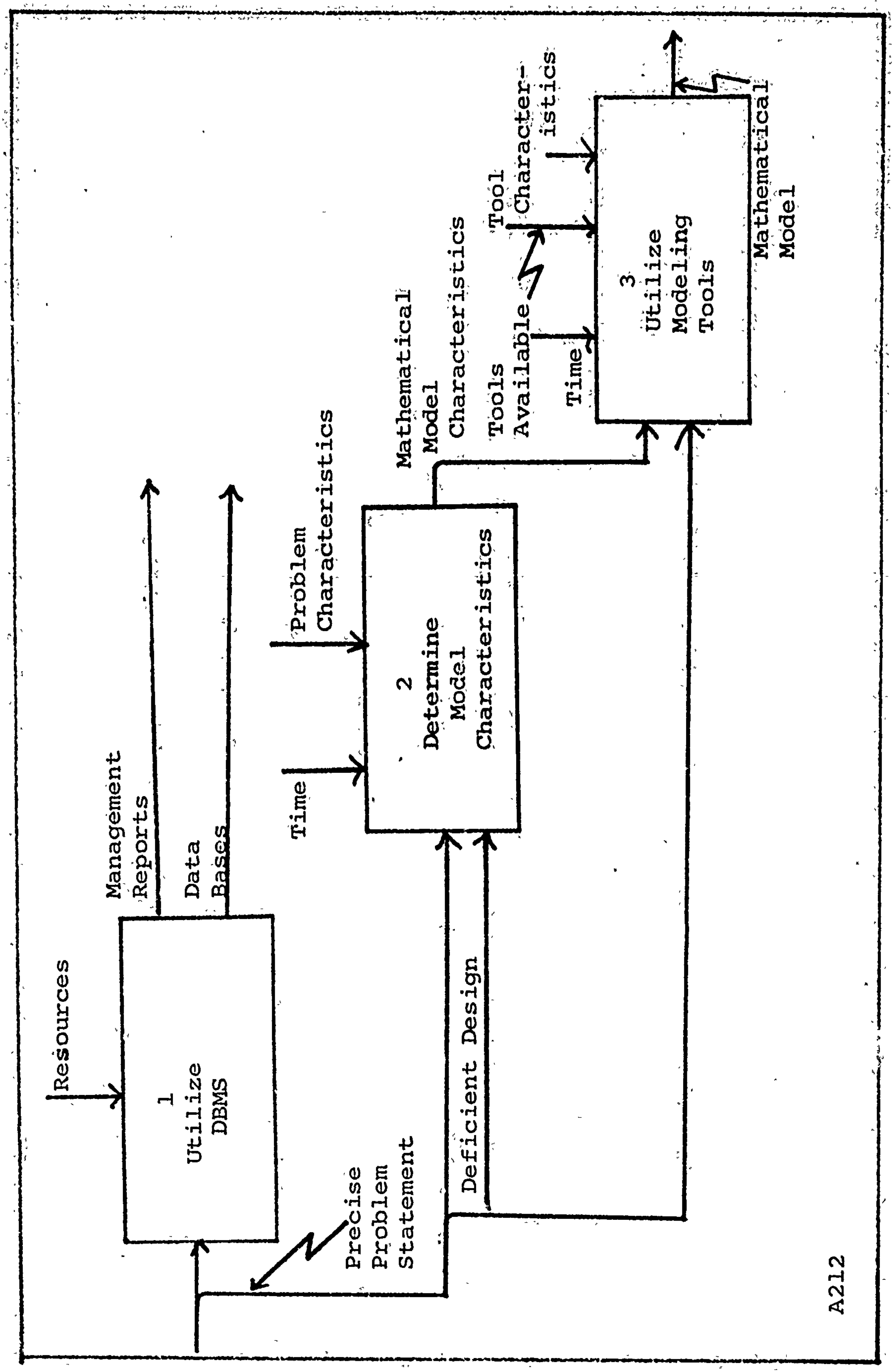
Utilize DBMS



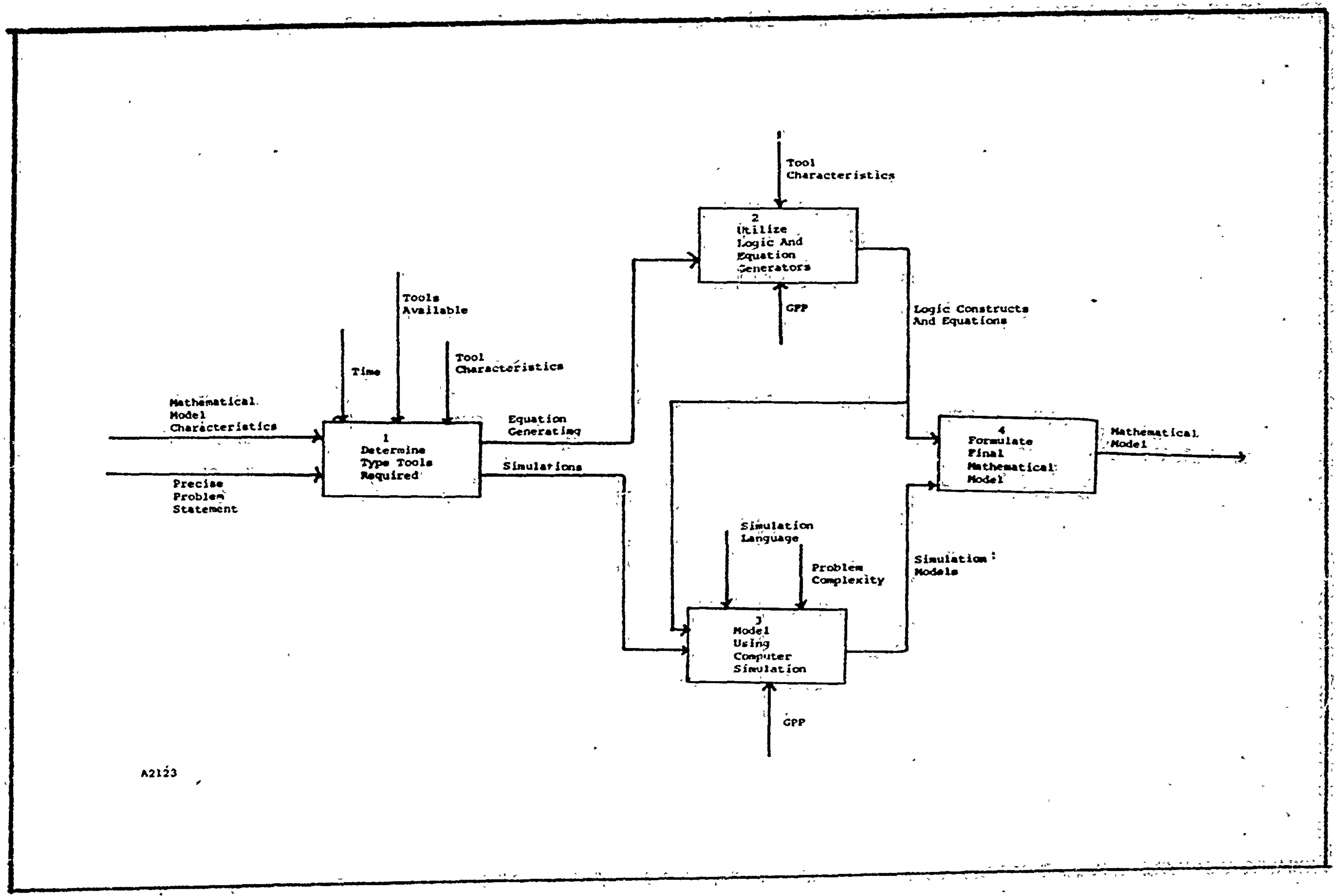
Utilize Software Design Tools


Problem Complexity
Time
2
Utilize Subgoals And Hillclimbing Techniques
Solution Approachcs/Techniques
Problem Complexity
Time
Solution Known
Problem Reduction
3
Utilize Working Backward Techniques
Solution Approaches/Techniques
CPP
Time
Problem Complexity
Time
Available Tools
Complete Valid System Requirements
1
Determine Type Tools Required
Solution Not Optimal
4
Apply Heuristic Techniques
Solution Approaches/Techniques
GPP
Optimal Solution
Time
Problem Complexity
5
Utilize Branch And Bound Techniques
Solution Approaches/ Techniques
GPP
Large System Problem
Simulation Accuracy
6
Utilize Simulations
Solution Approaches/ Techniques
GPP
STB
A2113

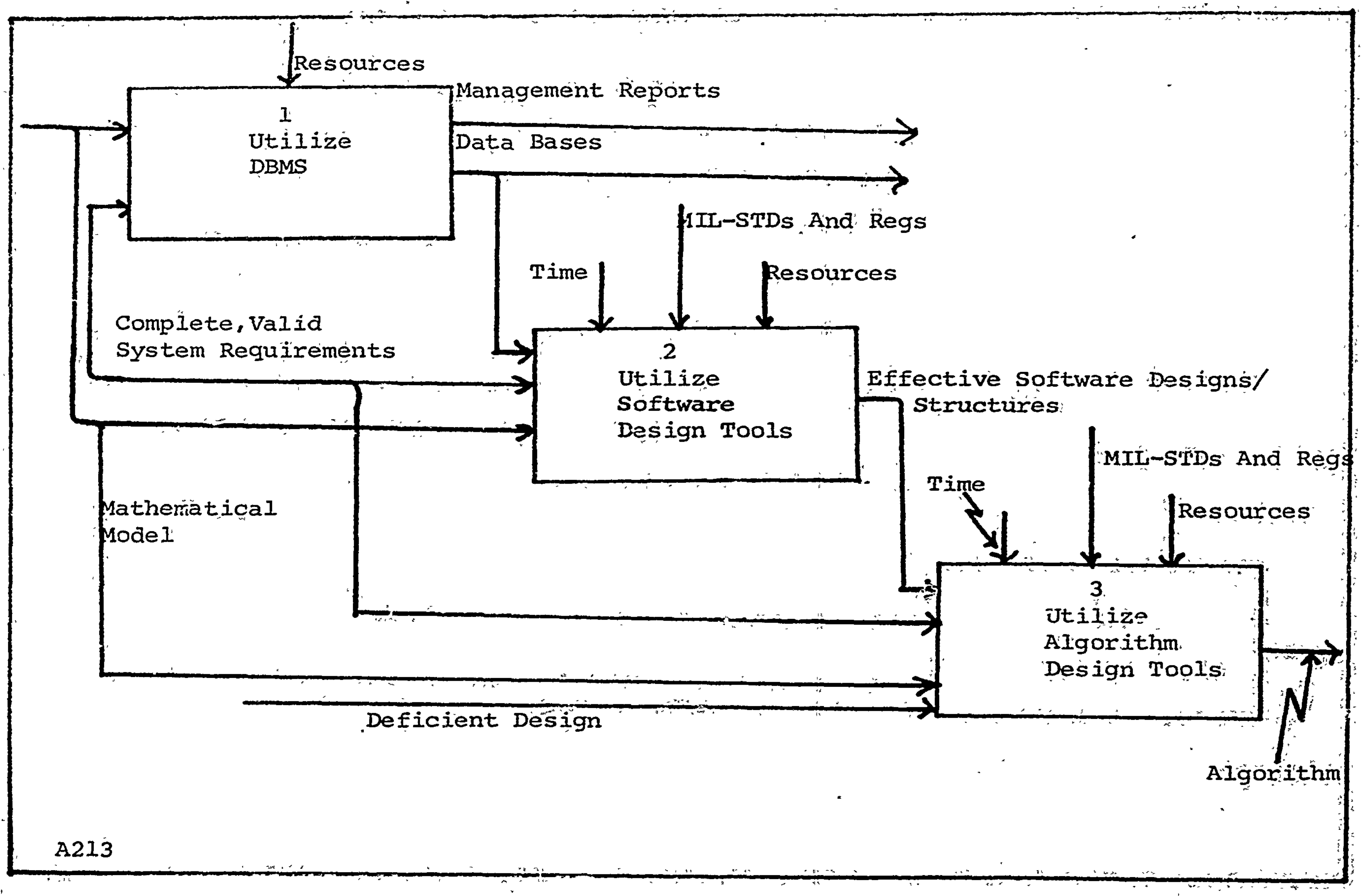
Utilize Algorithm Design Tools



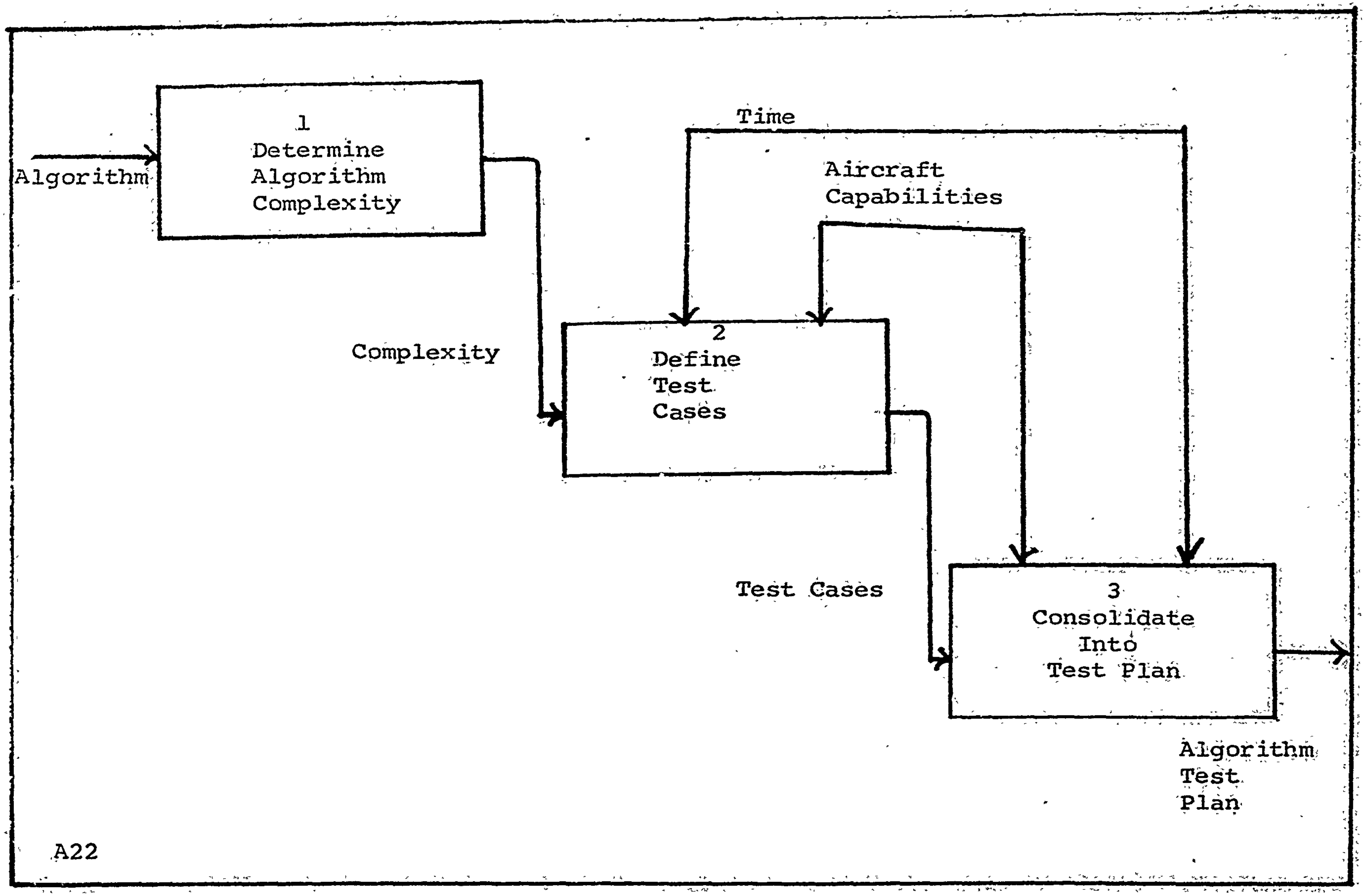
Develop Mathematical Model


Mathematical Model Characteristics
Precise Problem Statement
Time
Tools Available
Tool Characteristics
1 Determine Type Tools Required
Equation Generating
Simulations
Tool Characteristics
2 Utilize Logic And Equation Generators
GPP
Logic Constructs And Equations
Simulation Language
Problem Complexity
3 Model Using Computer Simulation
GPP
Simulation Models
4 Formulate Final Mathematical Model
Mathematical Model
A2123

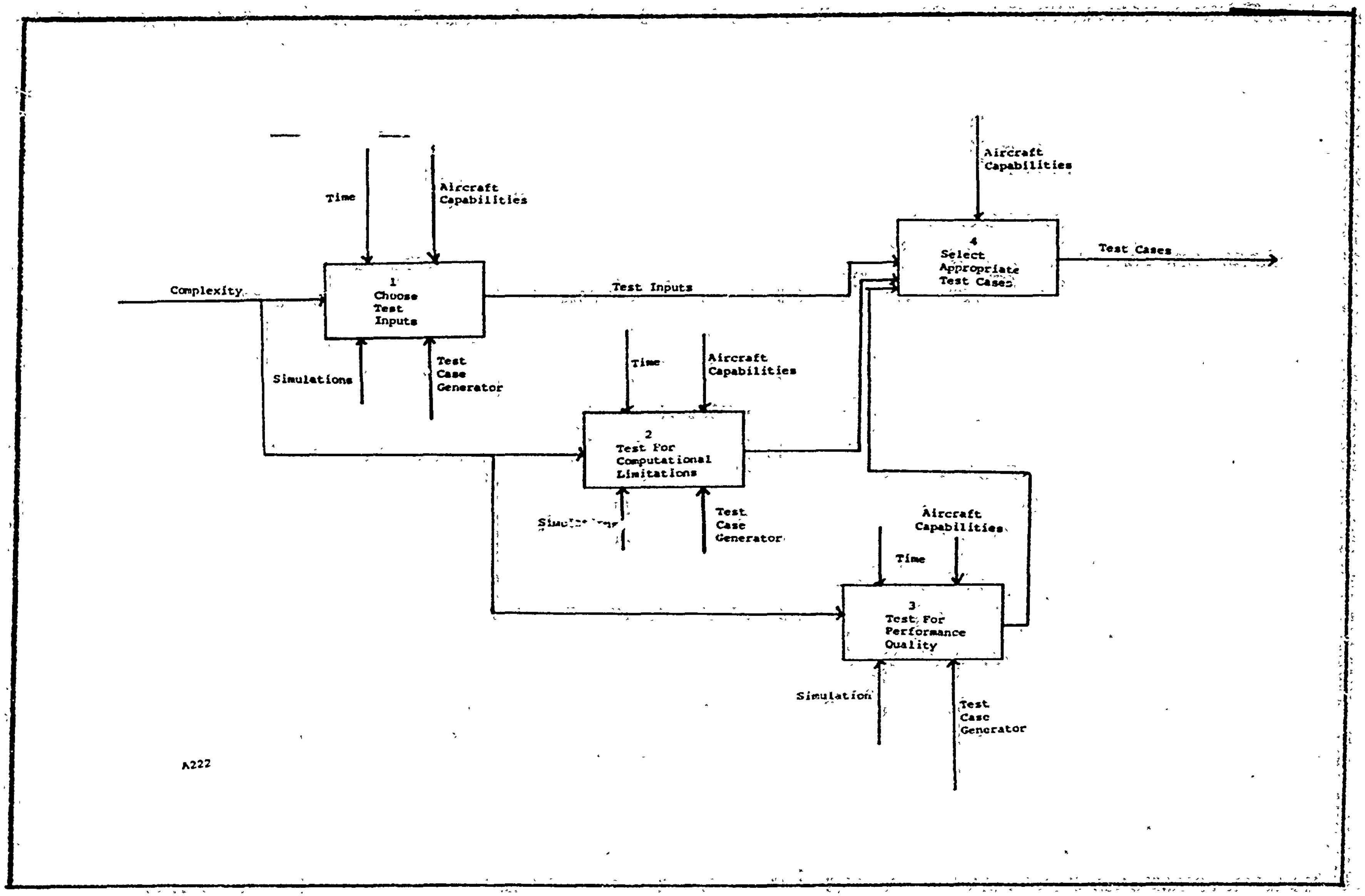
Utilize Modeling Tools



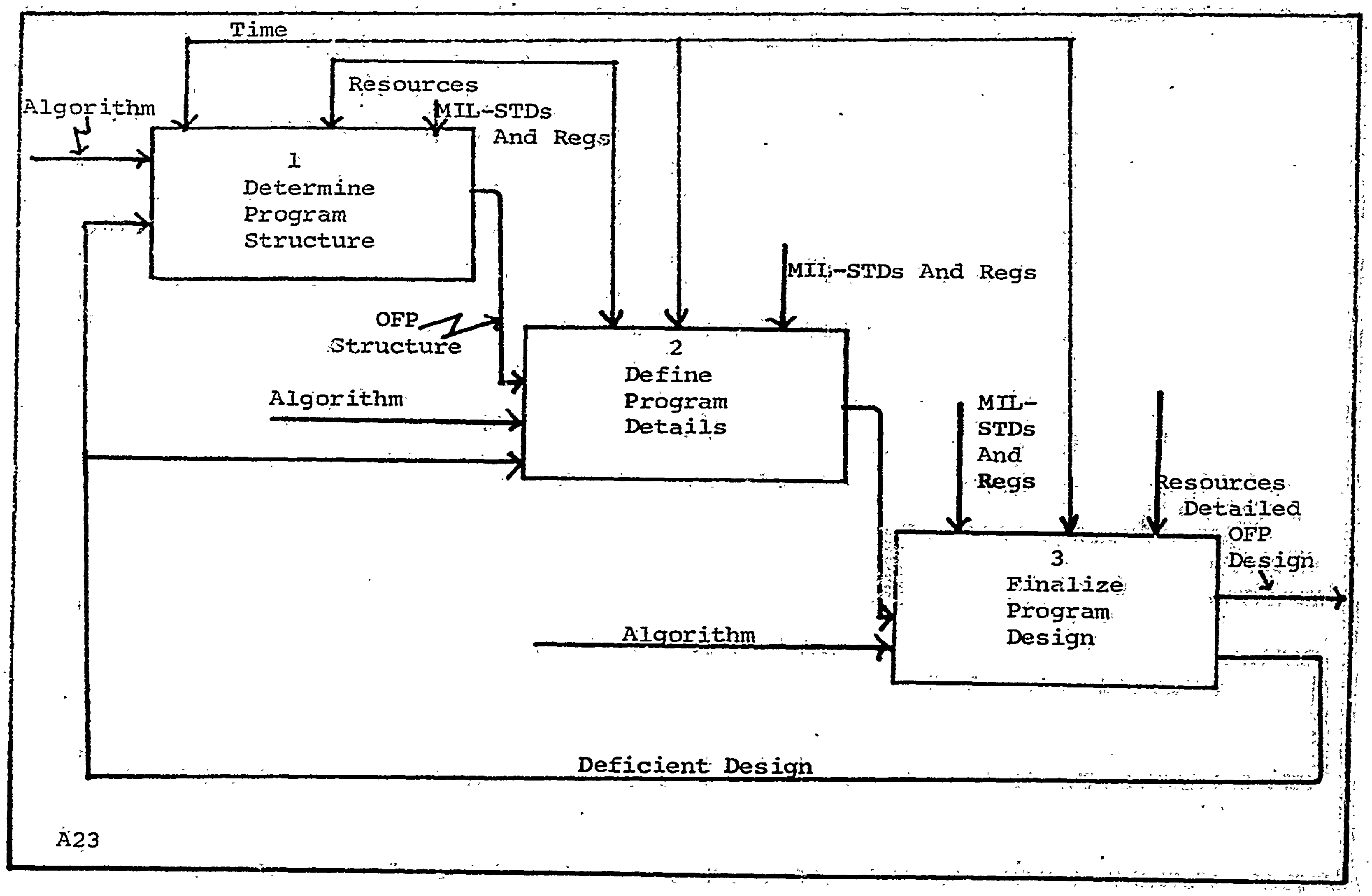
Design For Algorithm Effectiveness



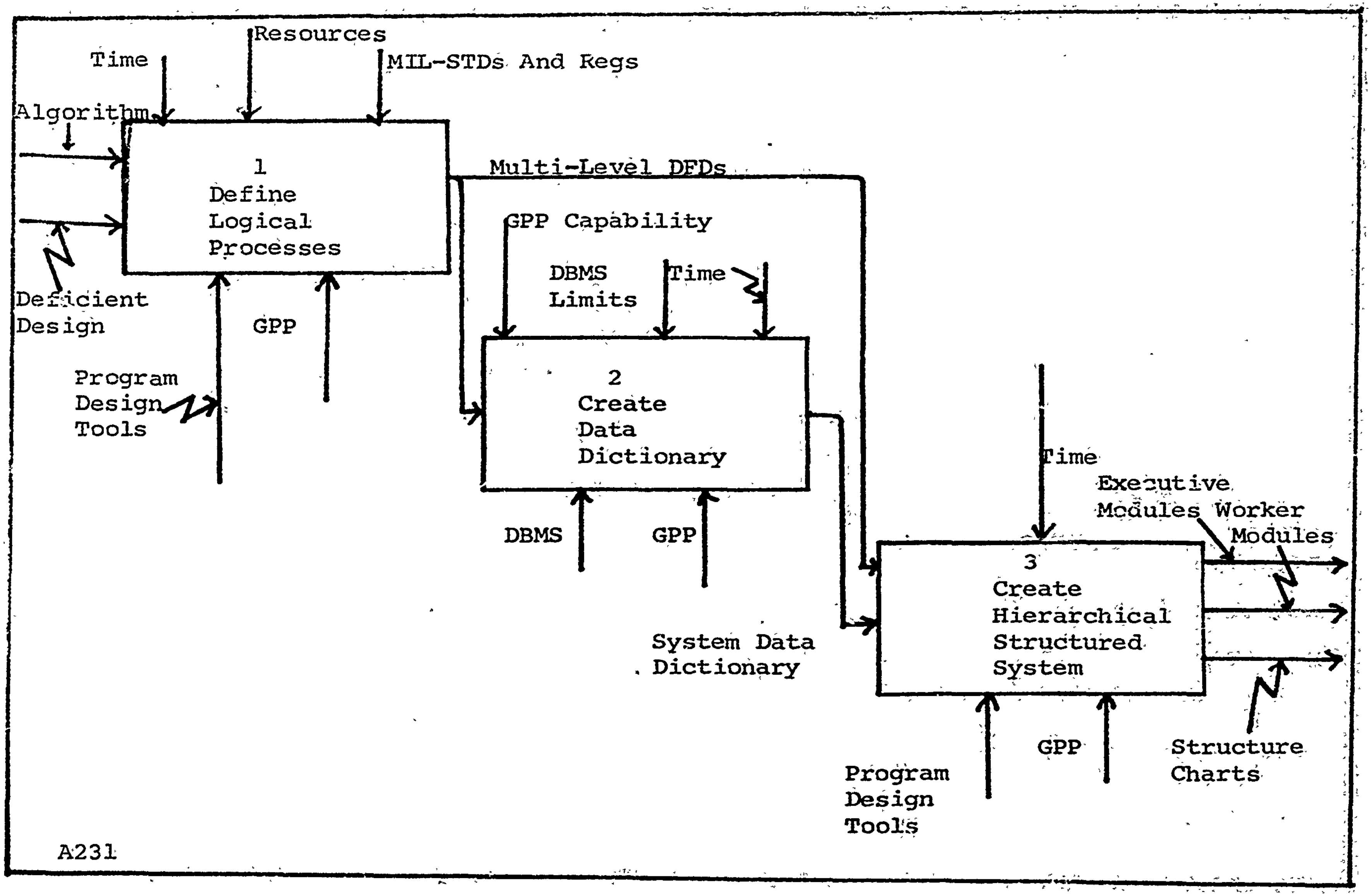
Develop Algorithm Test Plan



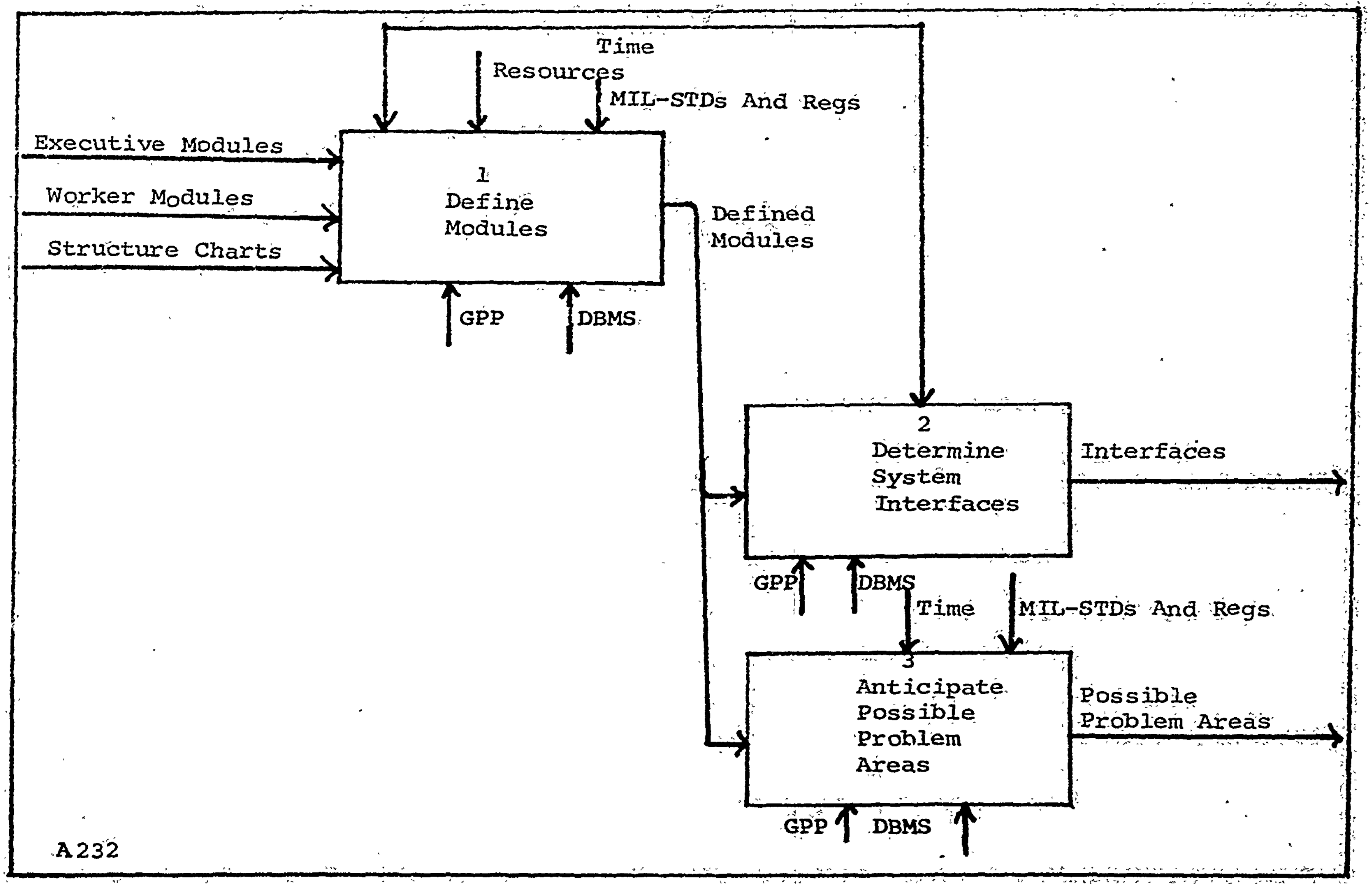
Define Test Cases



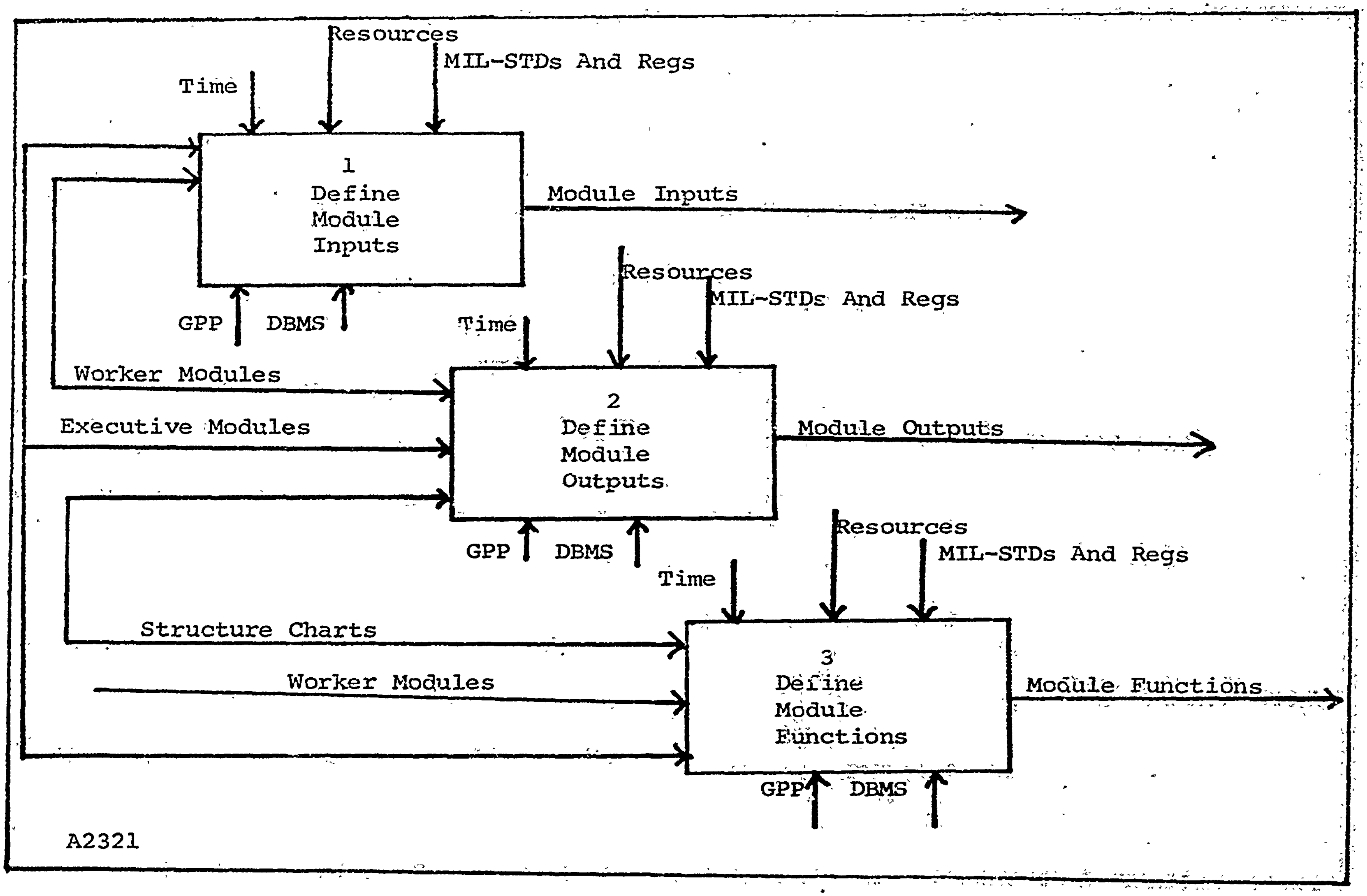
Design Program



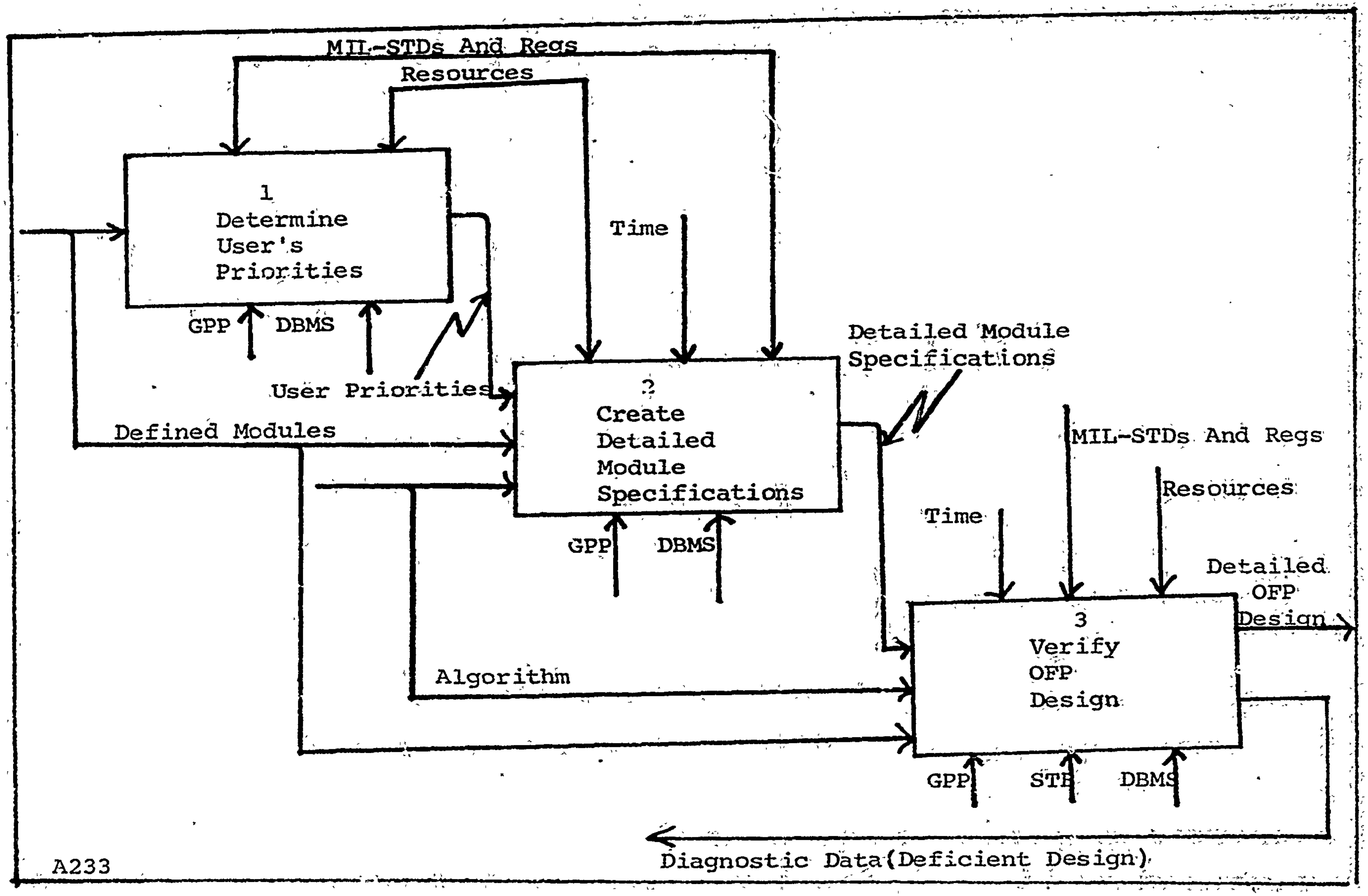
Determine Program Structure



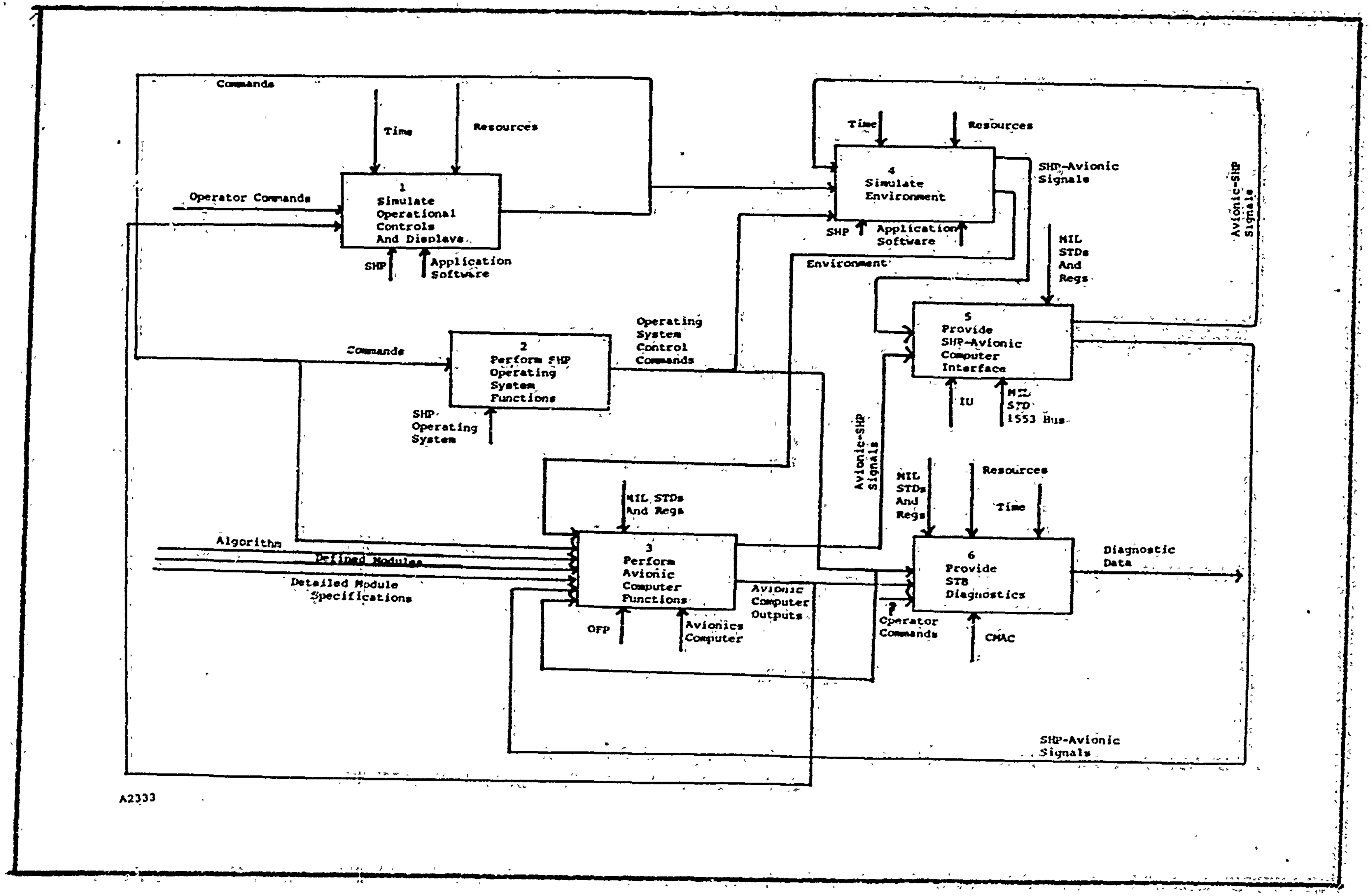
Define Program Details



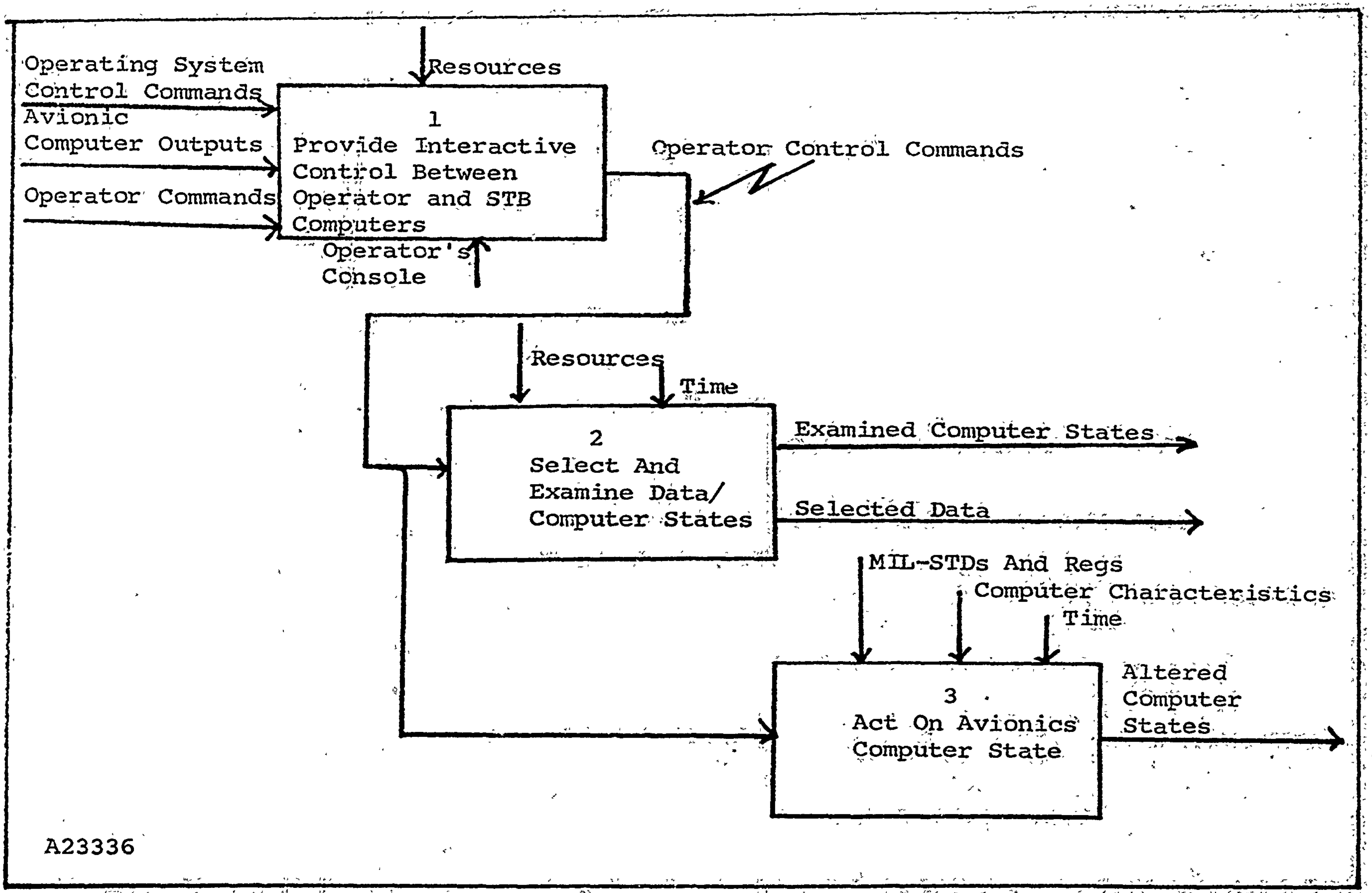
Define Modules



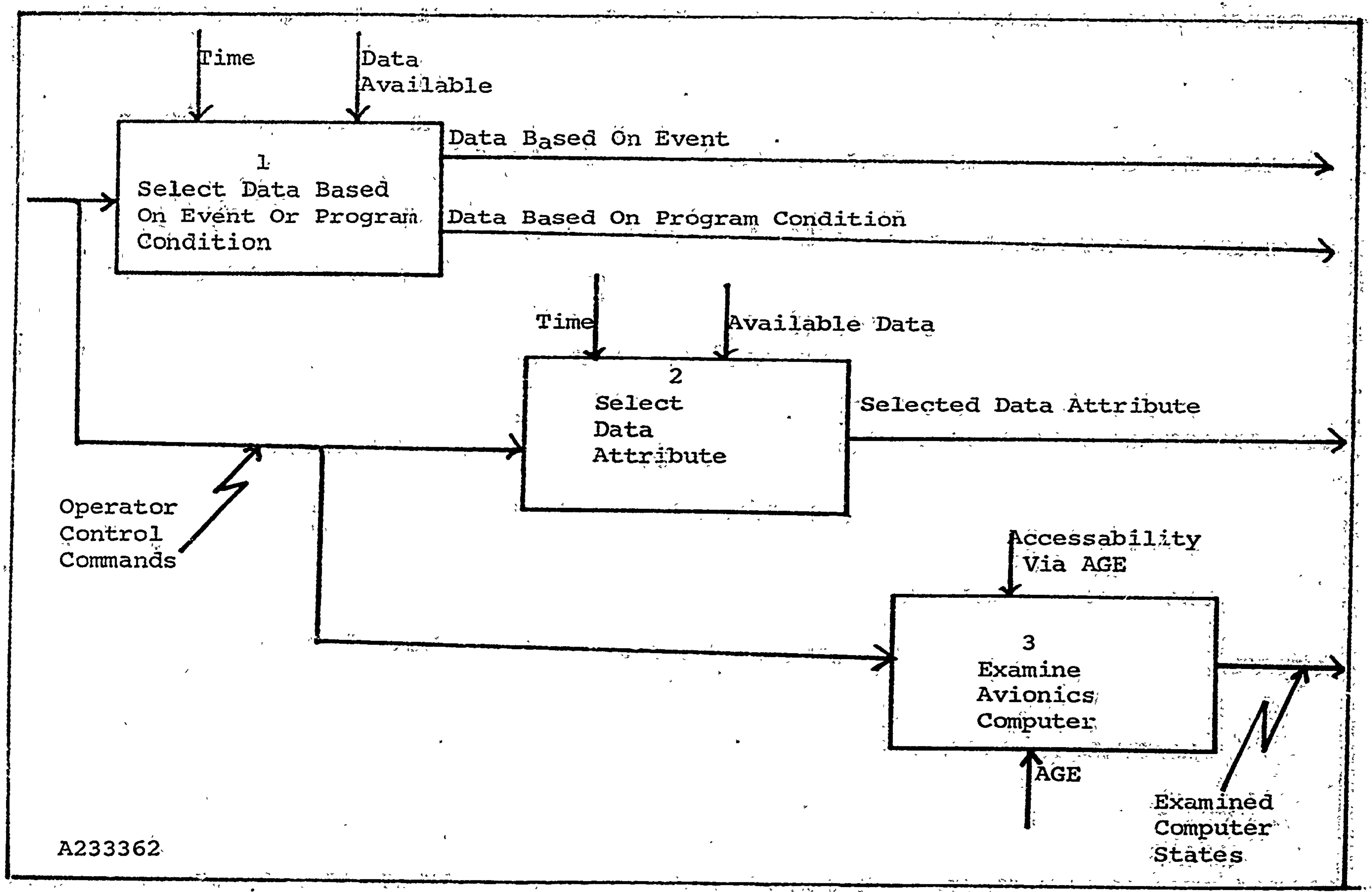
Finalize Program Design



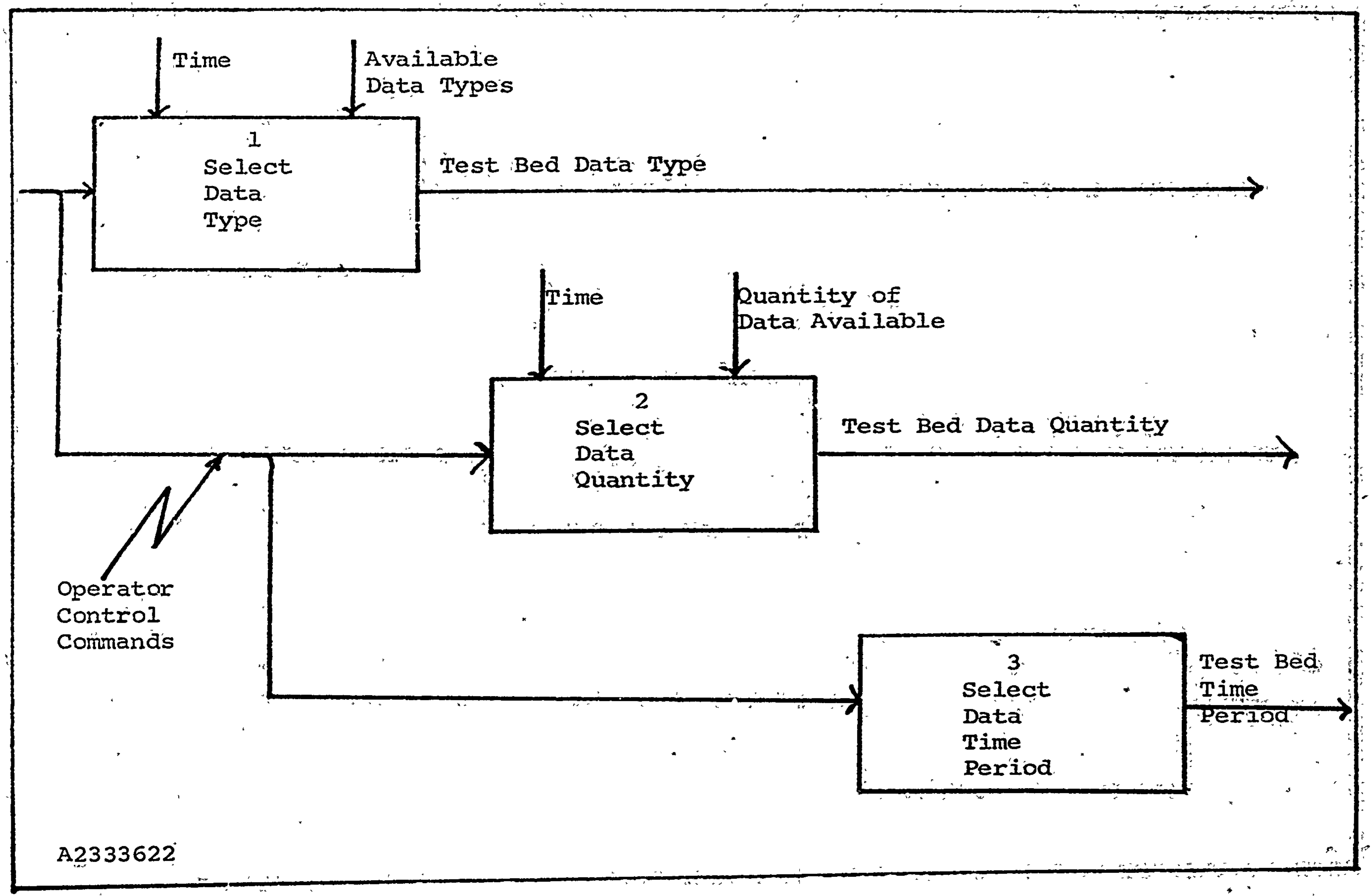
Verify OFP Design



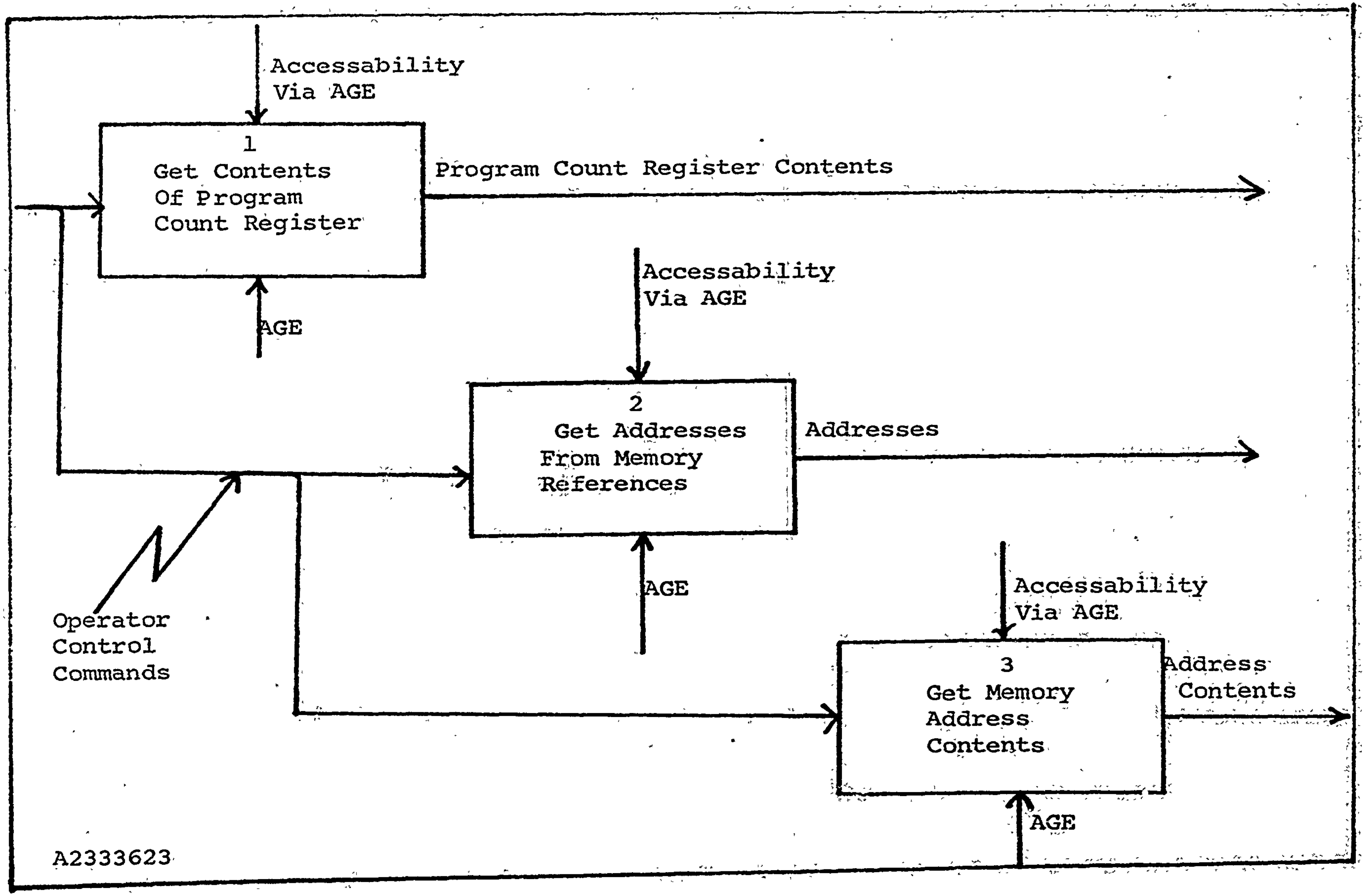
Provide STB Diagnostics



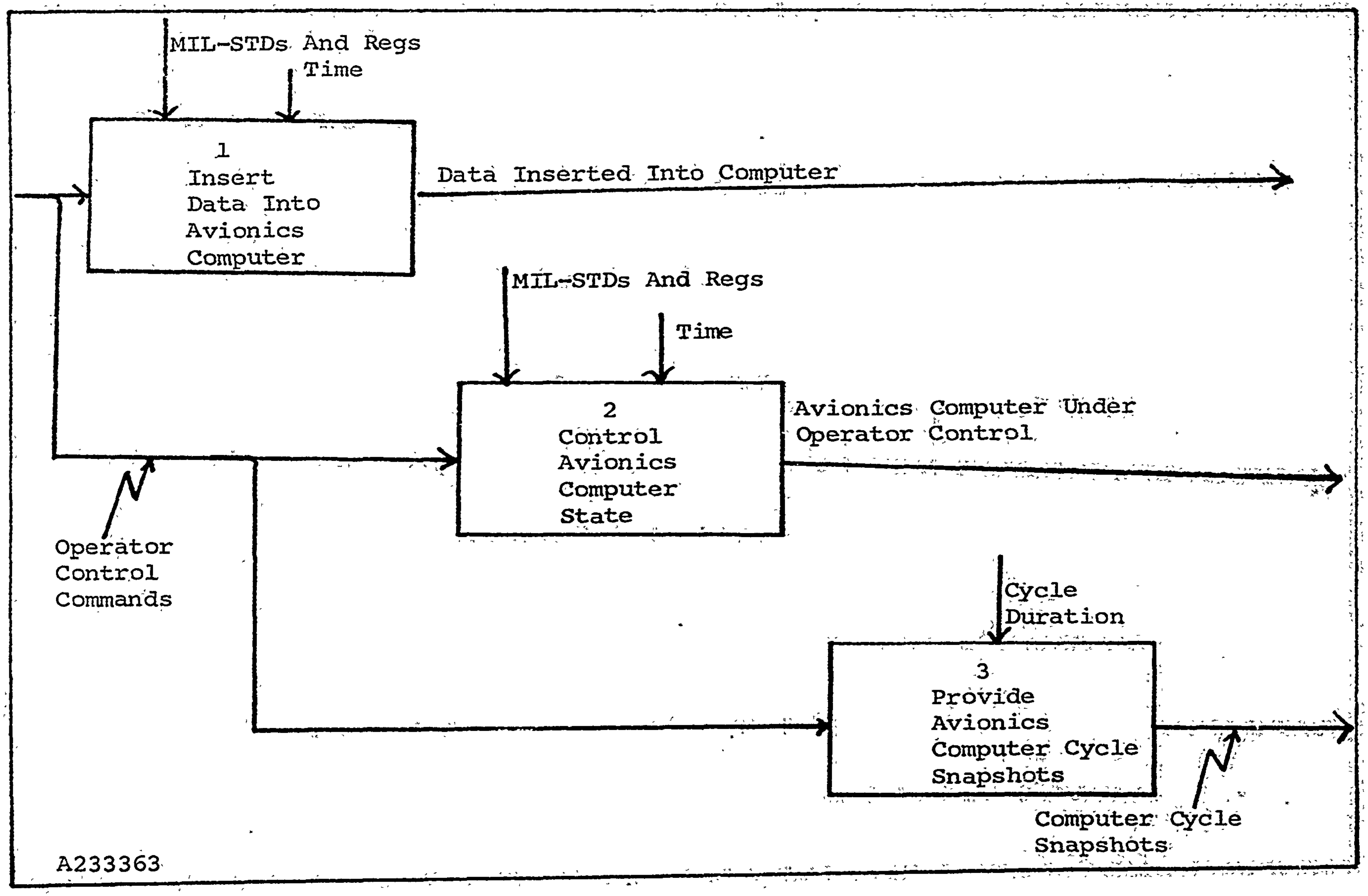
Select And Examine Data/Computer States


Time
Available Data Types
1 Select Data Type
Test Bed Data Type
Time
Quantity of Data Available
2 Select Data Quantity
Test Bed Data Quantity
Operator Control Commands
3 Select Data Time Period
Test Bed Time Period
A2333622

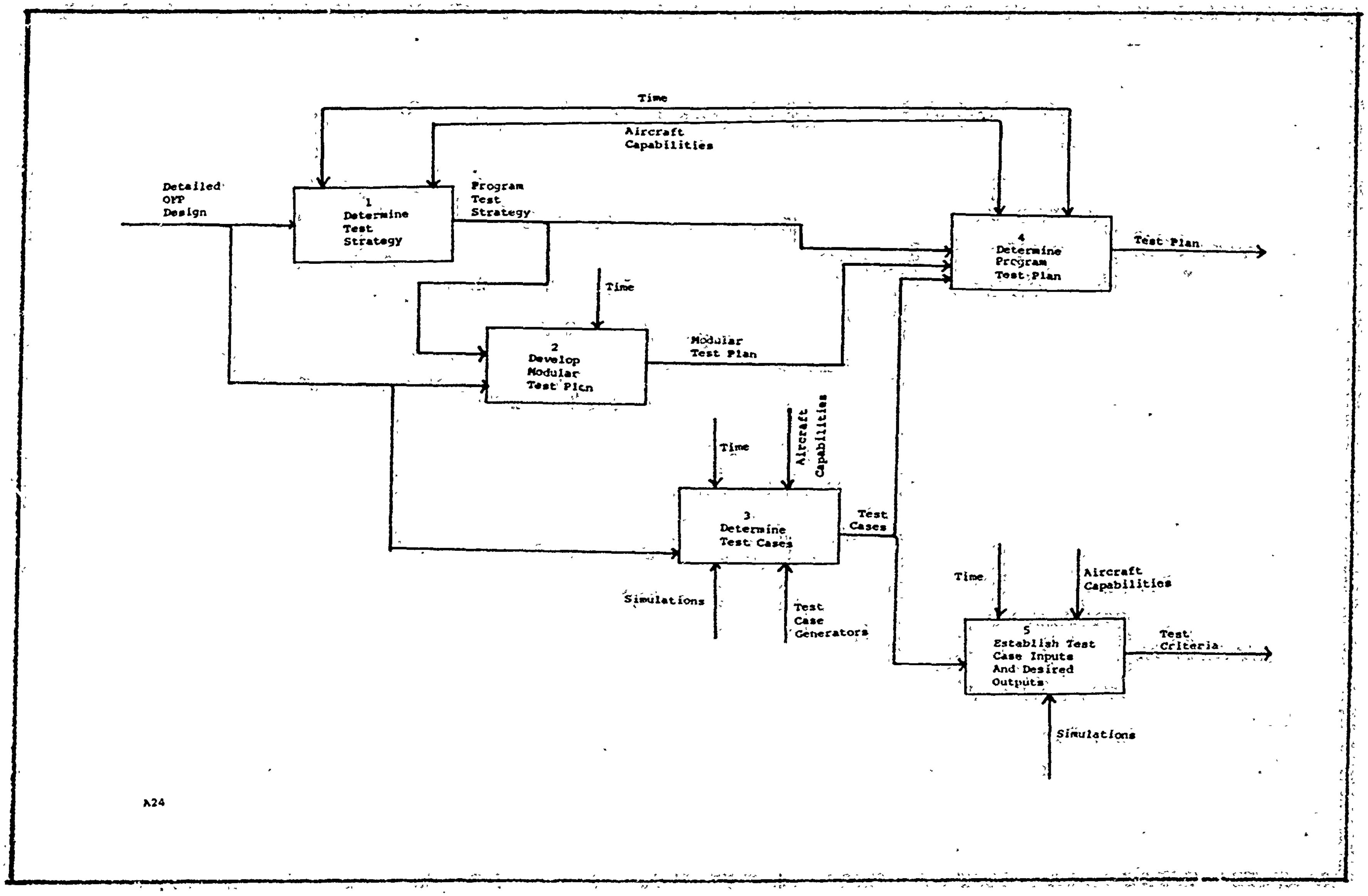
Select Data



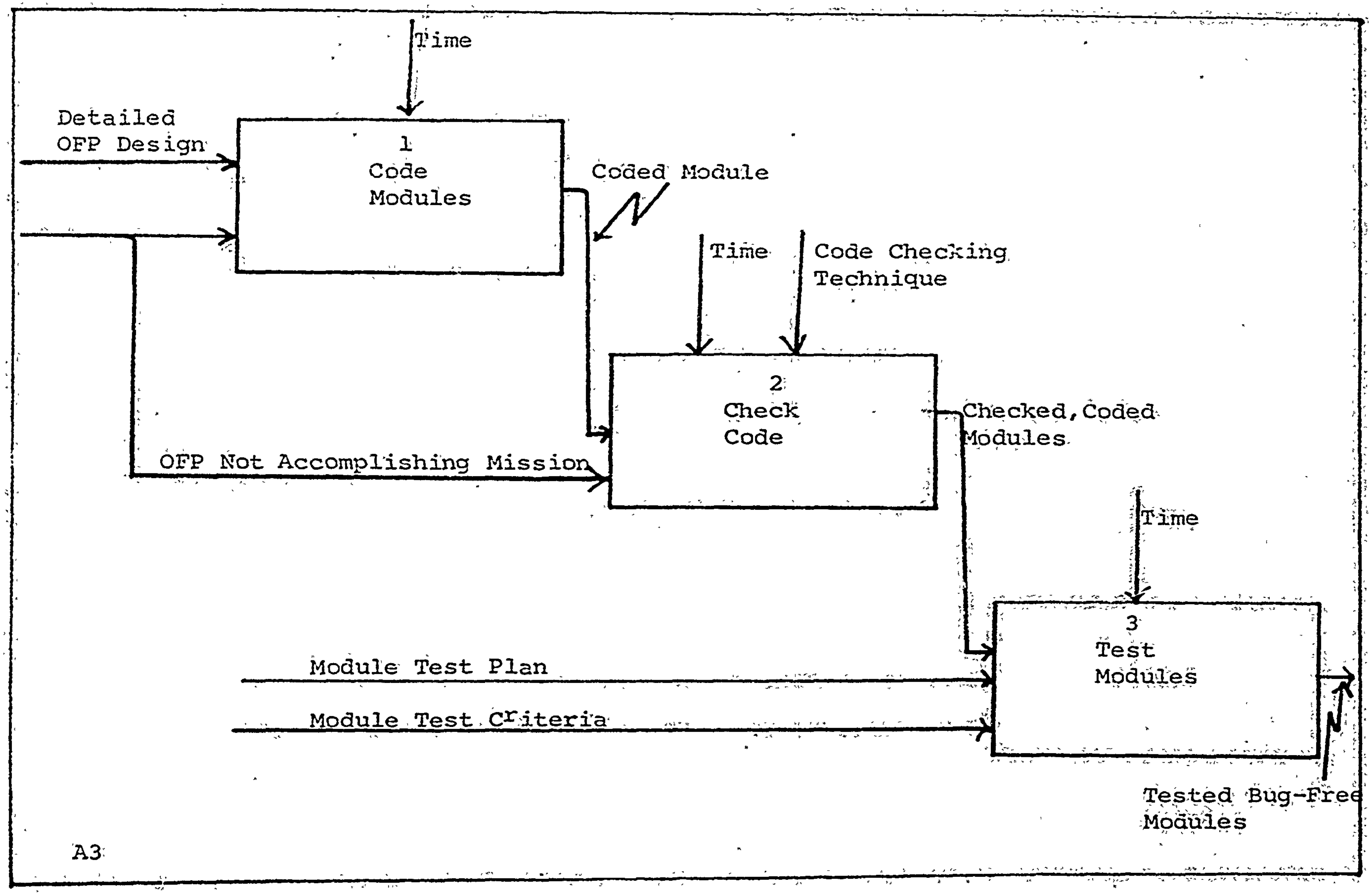
Examine Avionics Computer



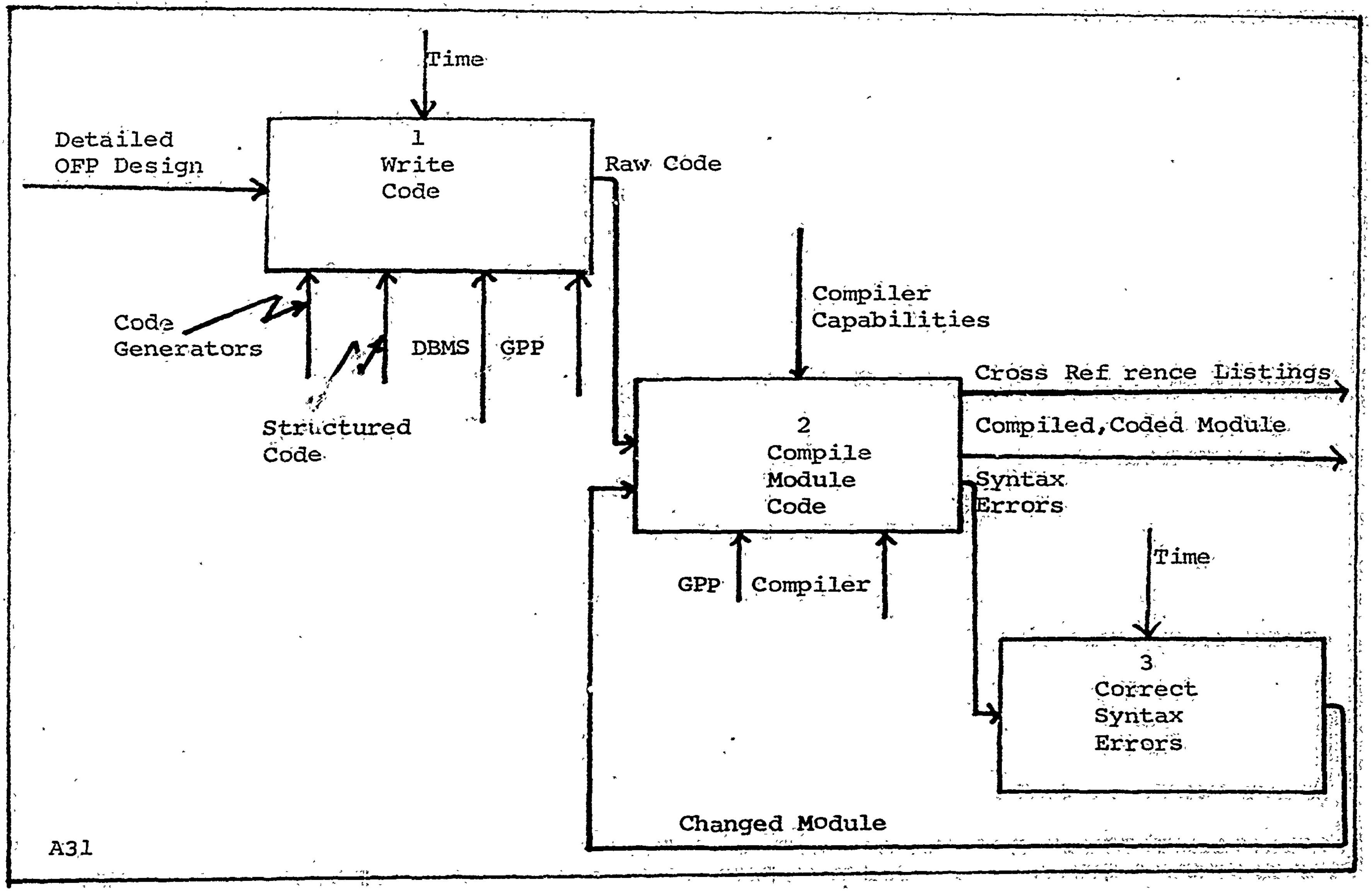
Act On Avionics Computer State



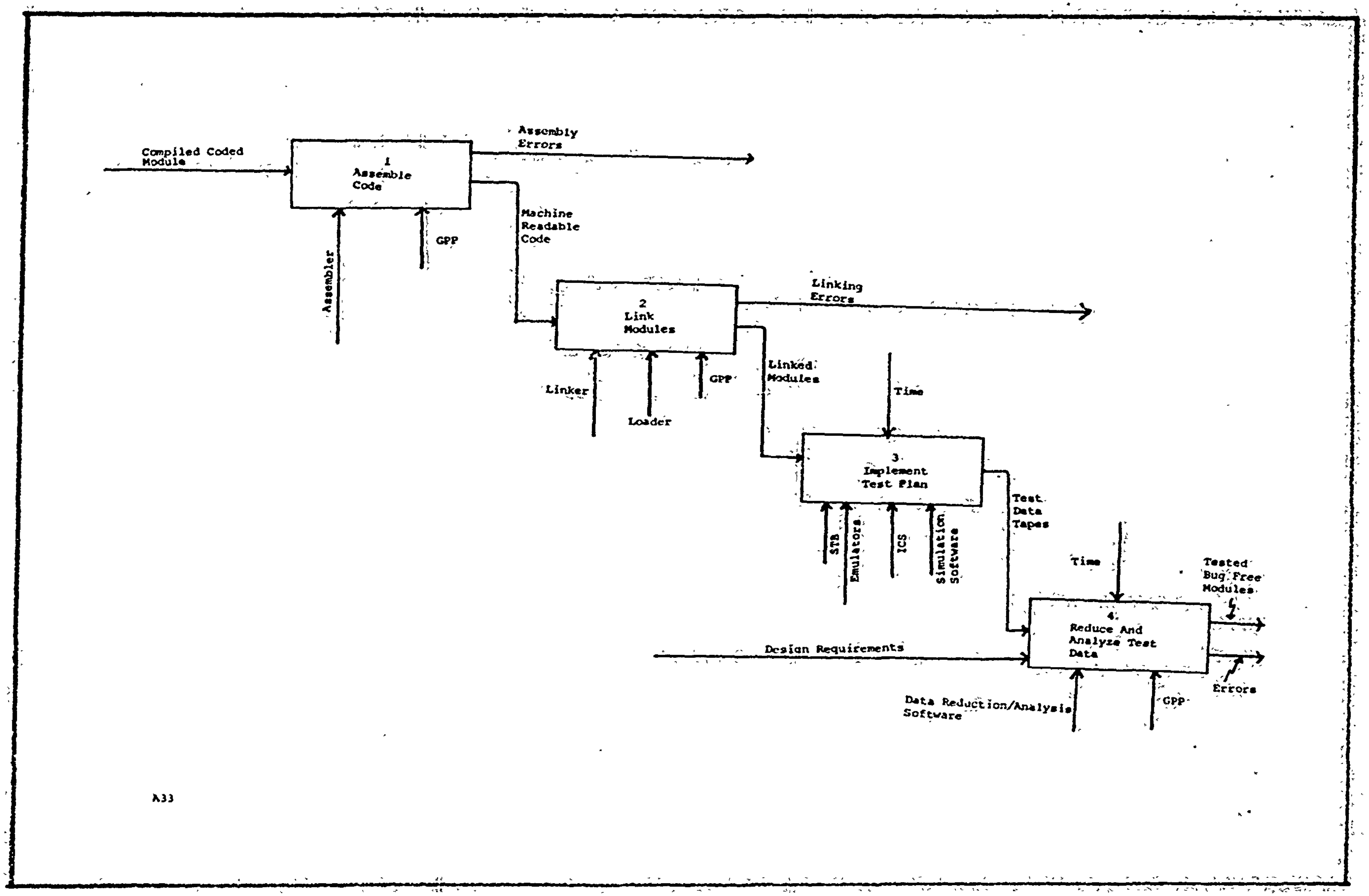
Develop Program Test Plan



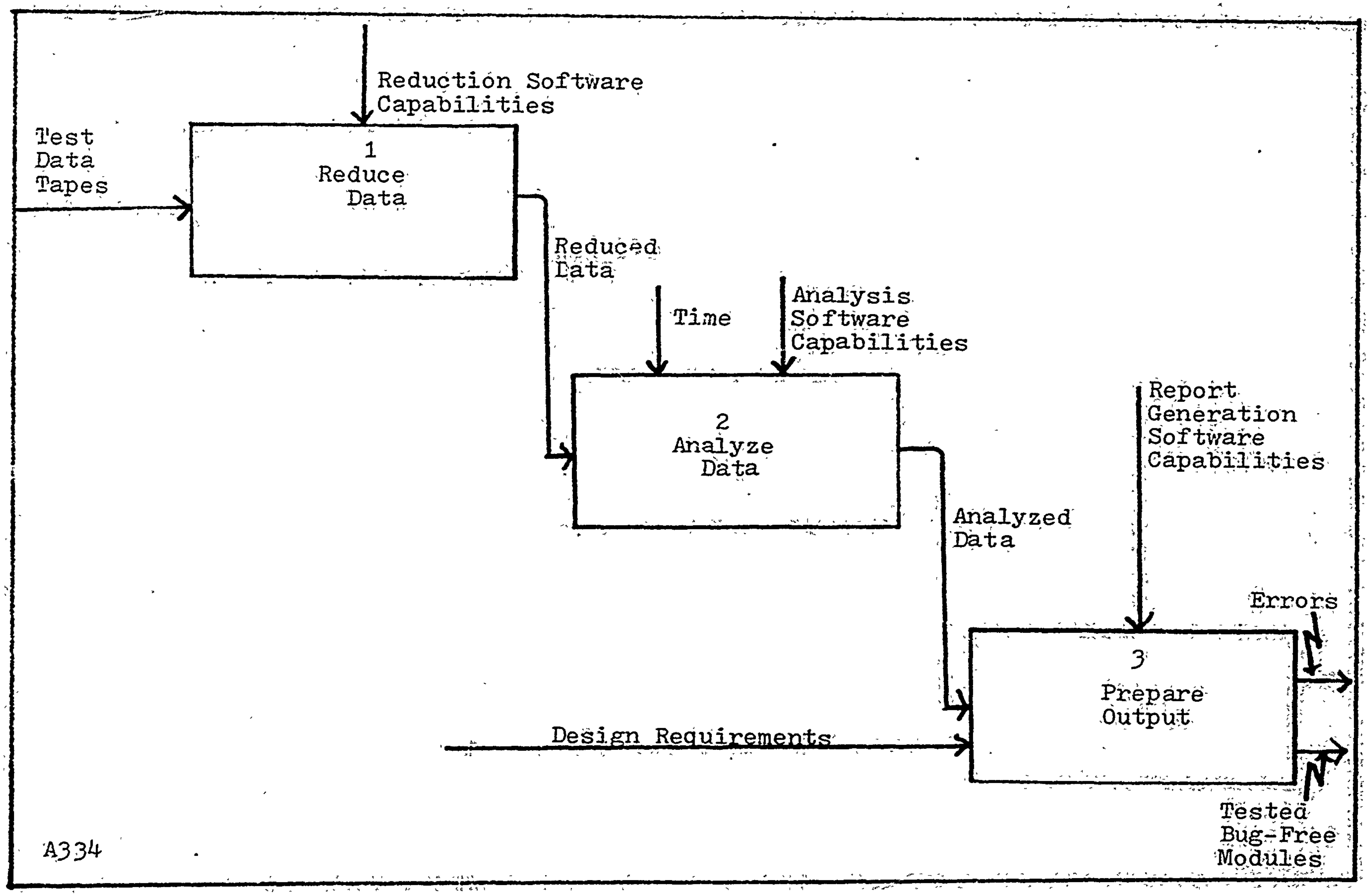
Code And Debug OFP



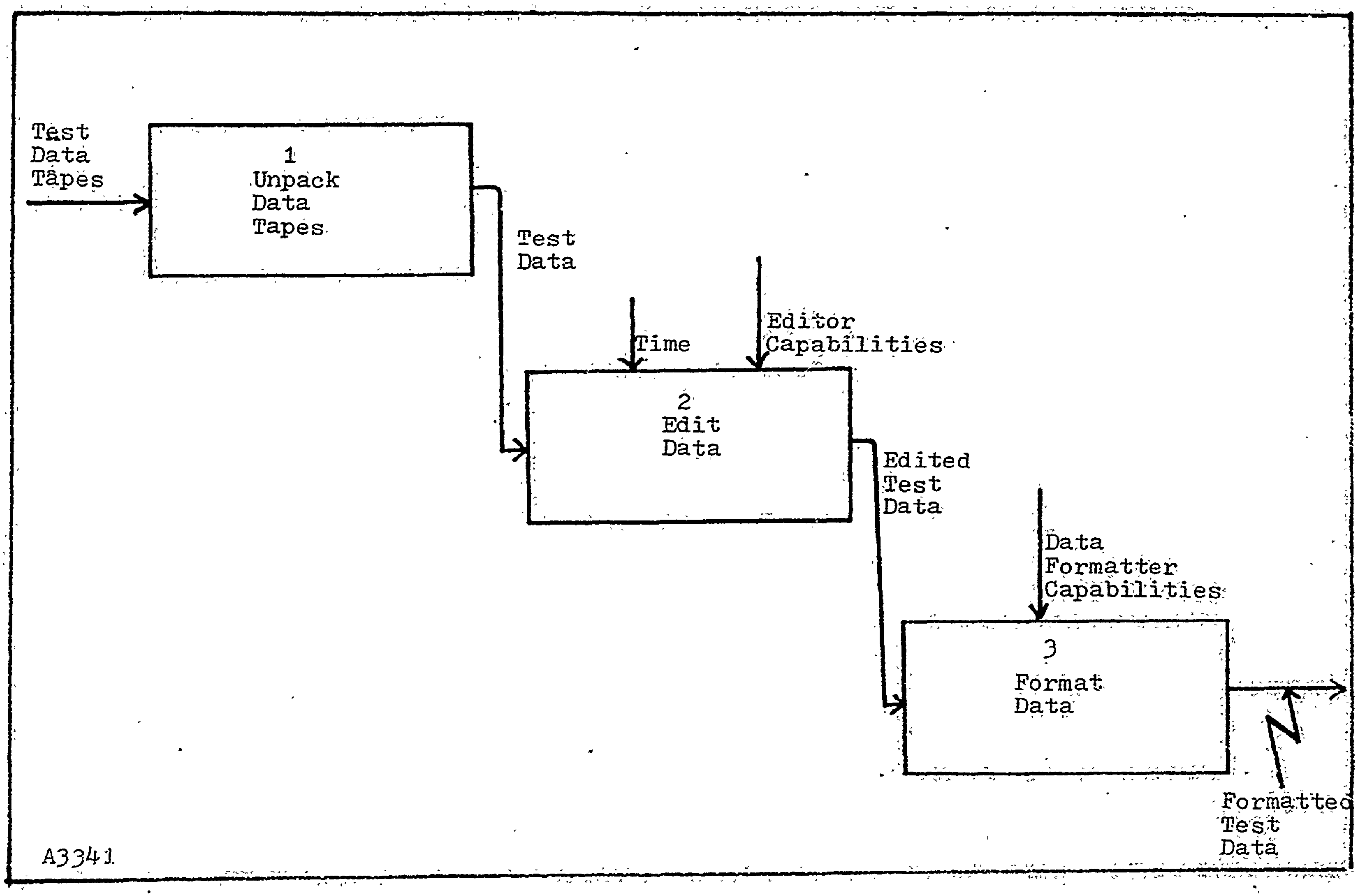
Code Modules



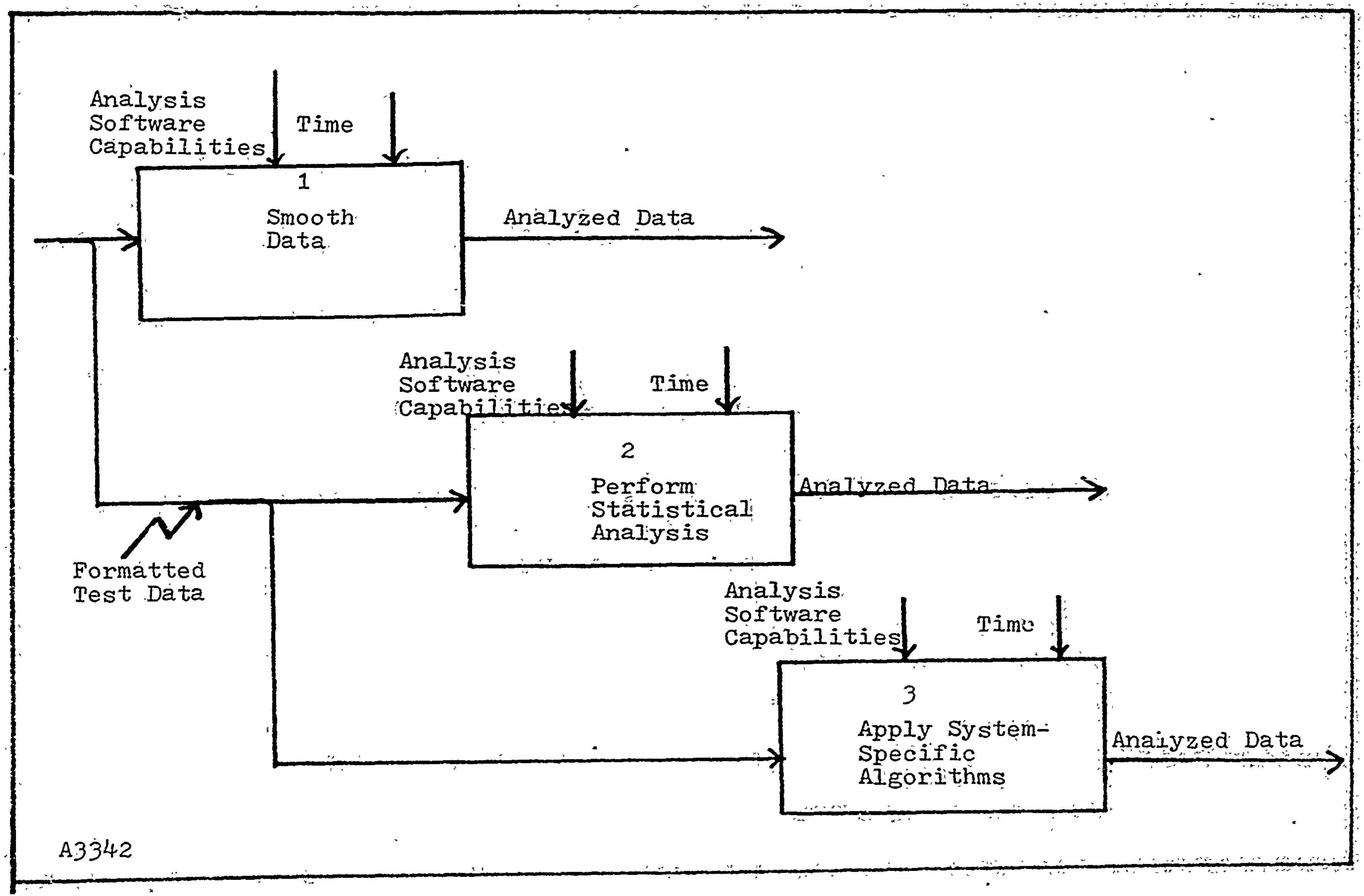
Test Modules



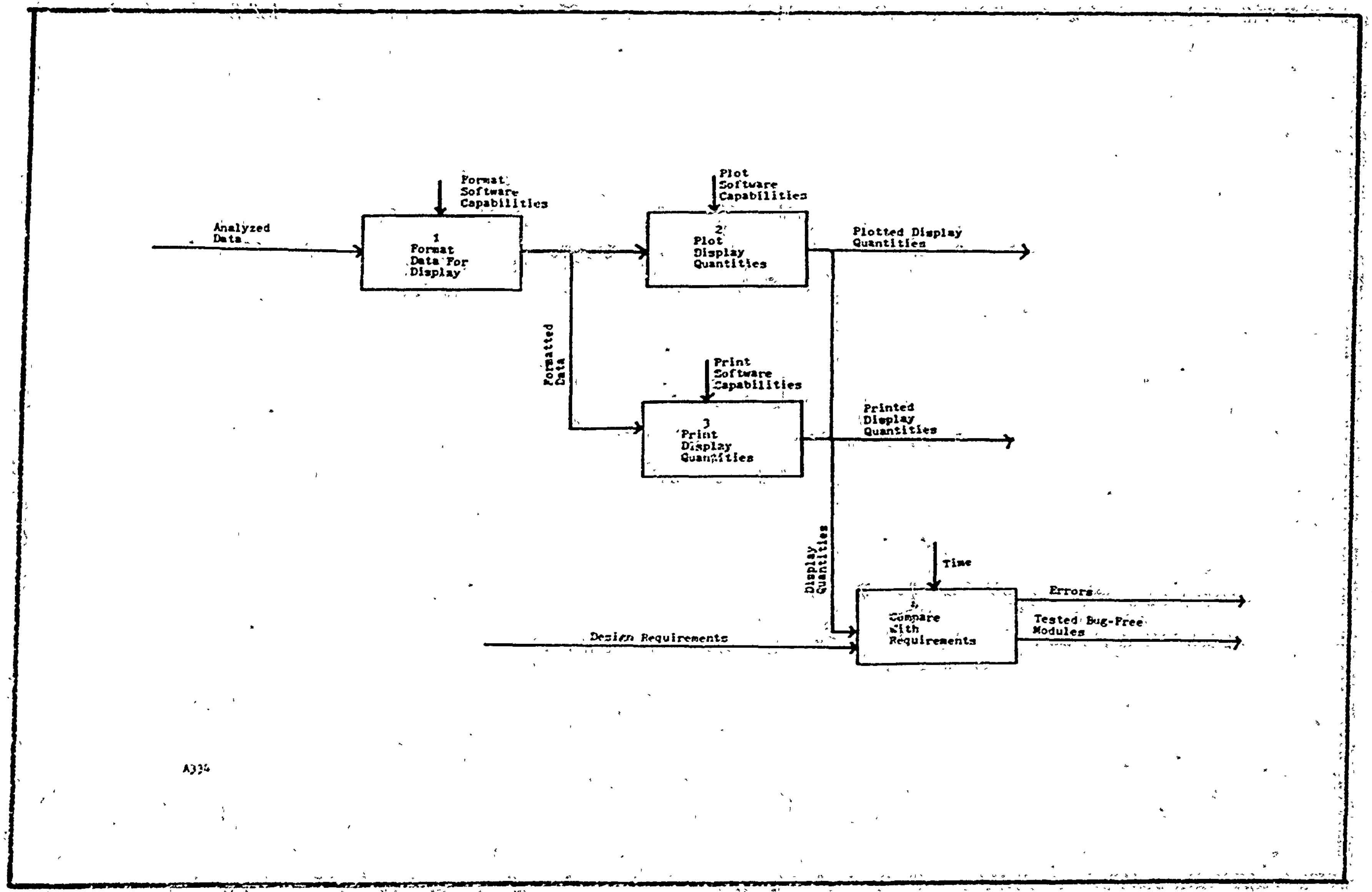
Reduce And Analyze Test Data


Test
Data
Tapes
1
Unpack
Data
Tapes
Test
Data
Time
Editor
Capabilities
2
Edit
Data
Edited
Test
Data
Data
Formatter
Capabilities
3
Format
Data
Formatted
Test
Data
A3341

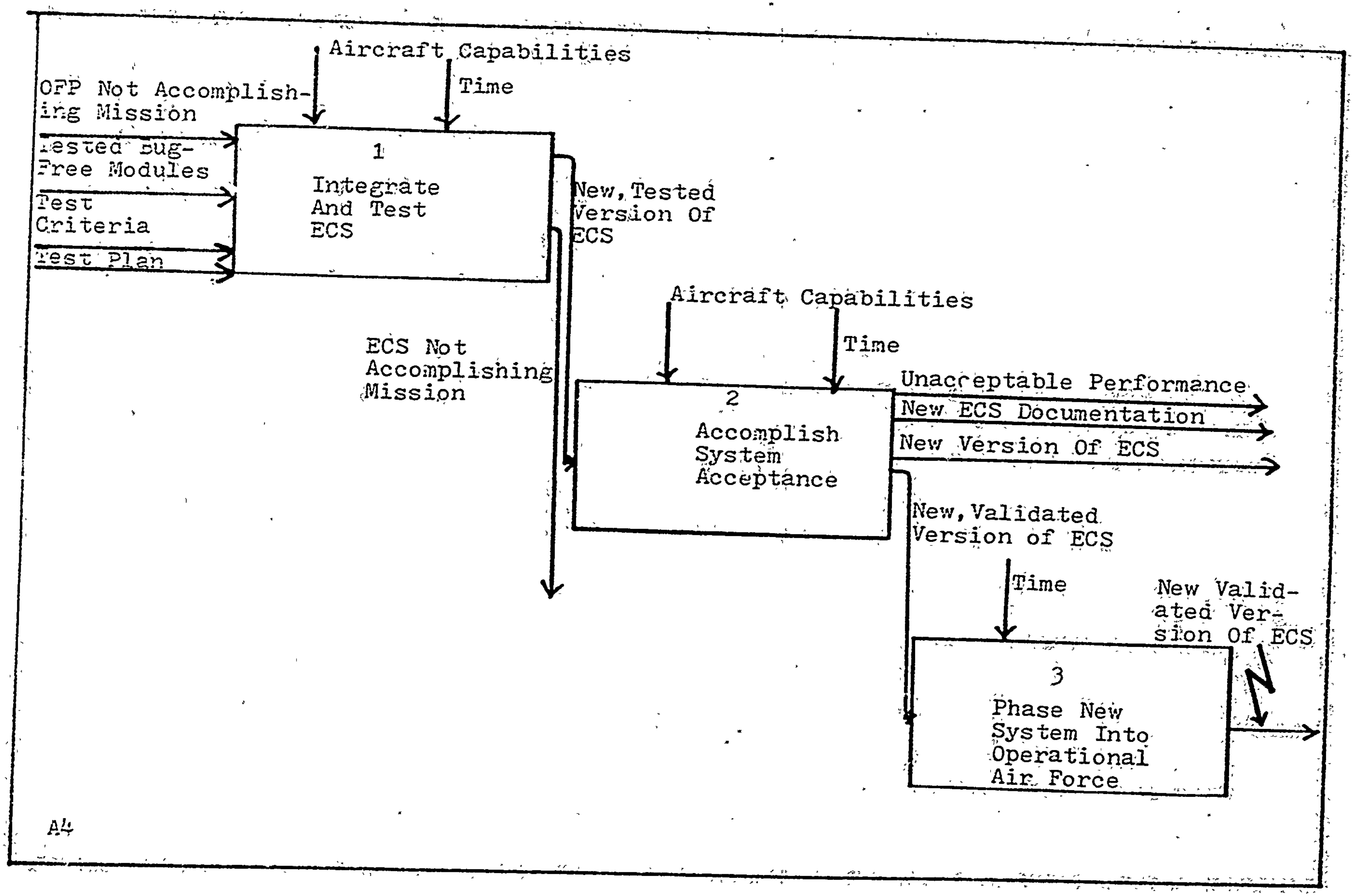
Reduce Data



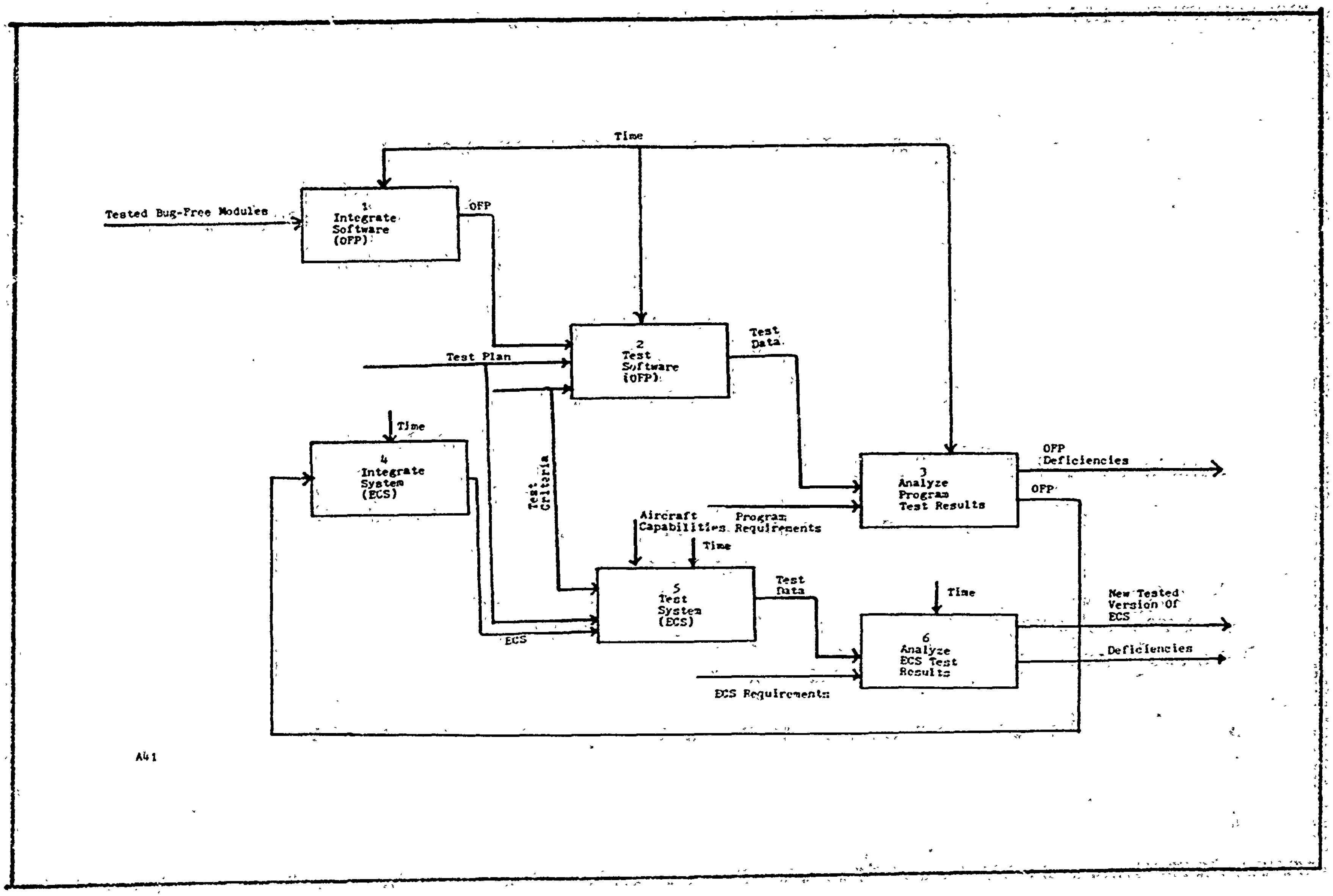
Analyze Data


Analyzed Data
Format Software Capabilities
1 Format Data For Display
Formatted Data
Plot Software Capabilities
2 Plot Display Quantities
Plotted Display Quantities
Print Software Capabilities
3 Print Display Quantities
Printed Display Quantities
Display Quantities
Time
4 Compare with Requirements
Errors
Tested Bug-Free Modules
Design Requirements
A334

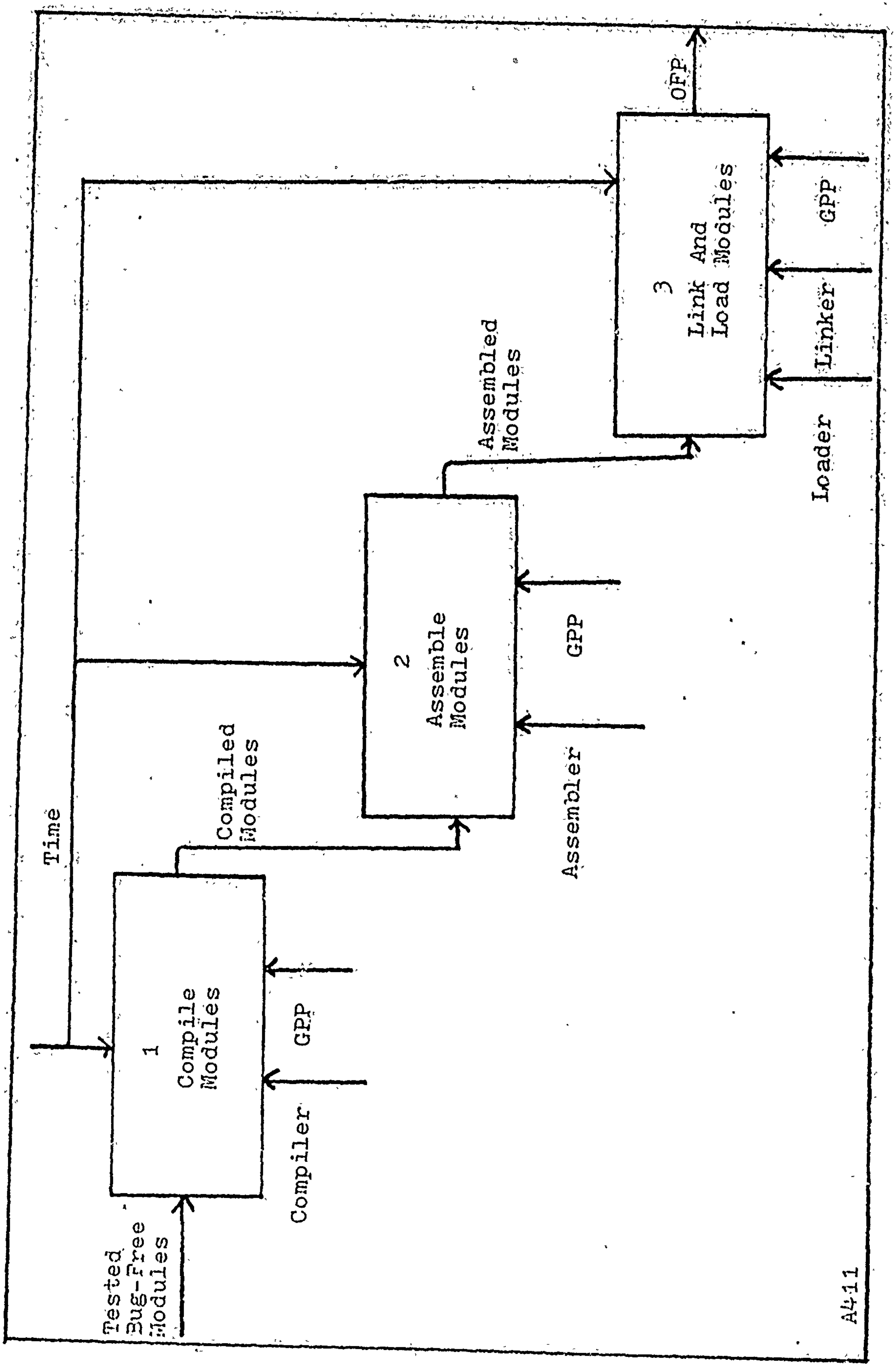
Prepare Output

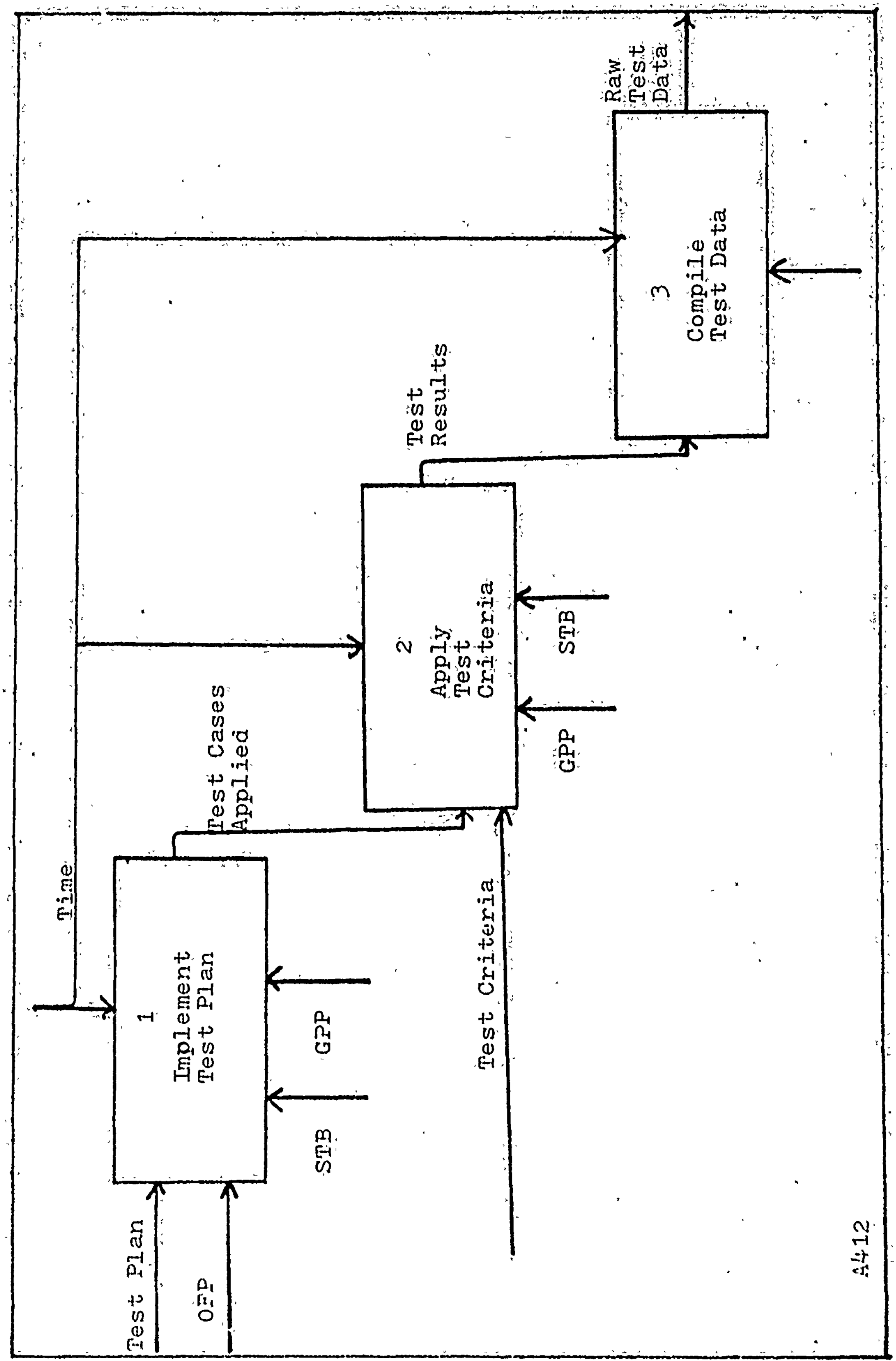

Aircraft Capabilities
Time
OFP Not Accomplishing Mission
Tested Bug-Free Modules
Test Criteria
Test Plan
1
Integrate And Test ECS
New, Tested Version Of ECS
ECS Not Accomplishing Mission
Aircraft Capabilities
Time
2
Accomplish System Acceptance
Unacceptable Performance
New ECS Documentation
New Version Of ECS
New, Validated Version of ECS
Time
3
Phase New System Into Operational Air Force
New Validated Version Of ECS
A4

Test And Verify OFP



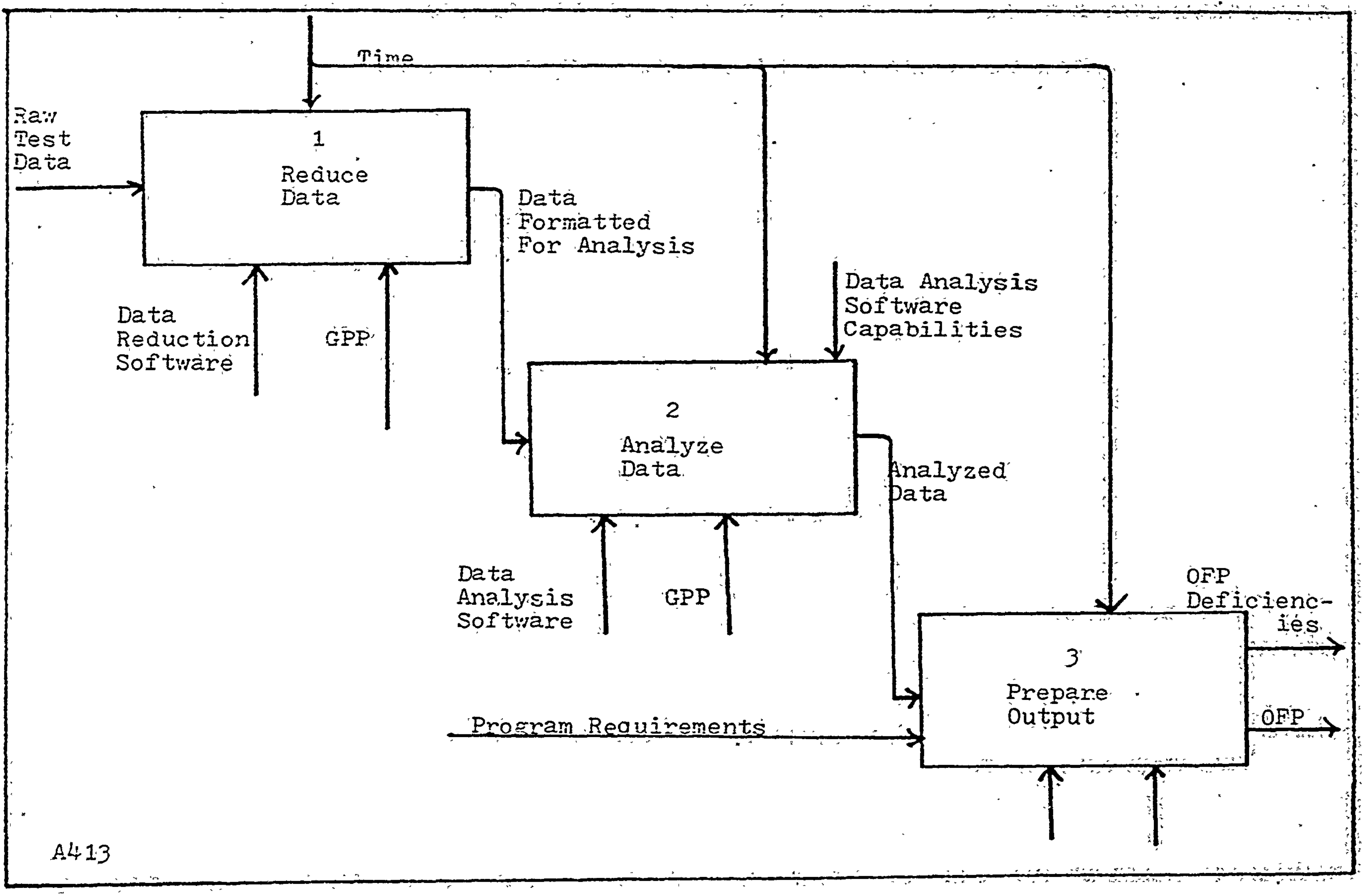
Integrate And Test ECS


Time
1
Compile
Modules
Tested
Bug-Free
Modules
GPP
Compiler
Compiled
Modules
2
Assemble
Modules
GPP
Assembler
Assembled
Modules
3
Link And
Load Modules
OFP
GPP
Linker
Loader
A411
Integrate Software

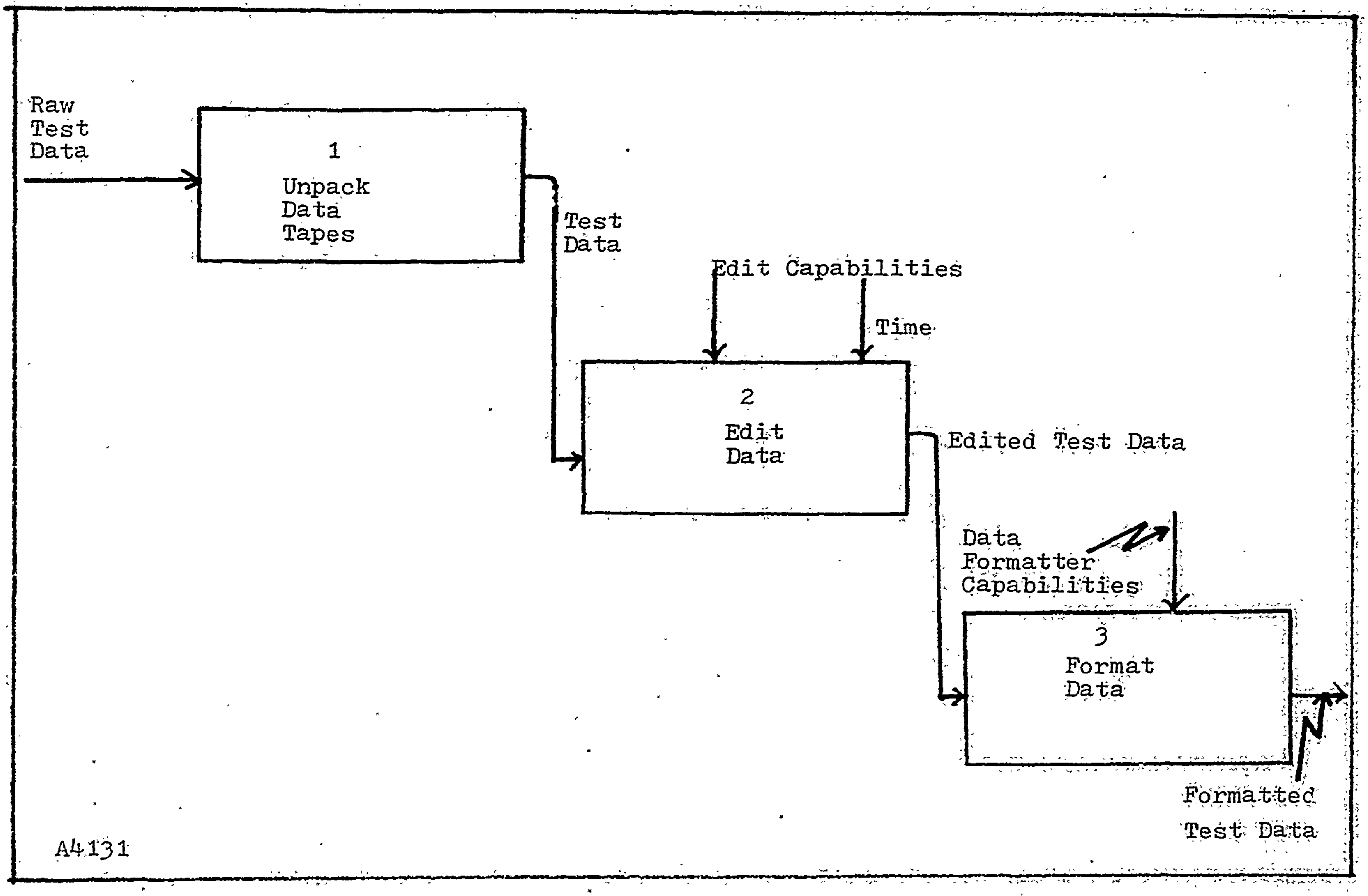


A412

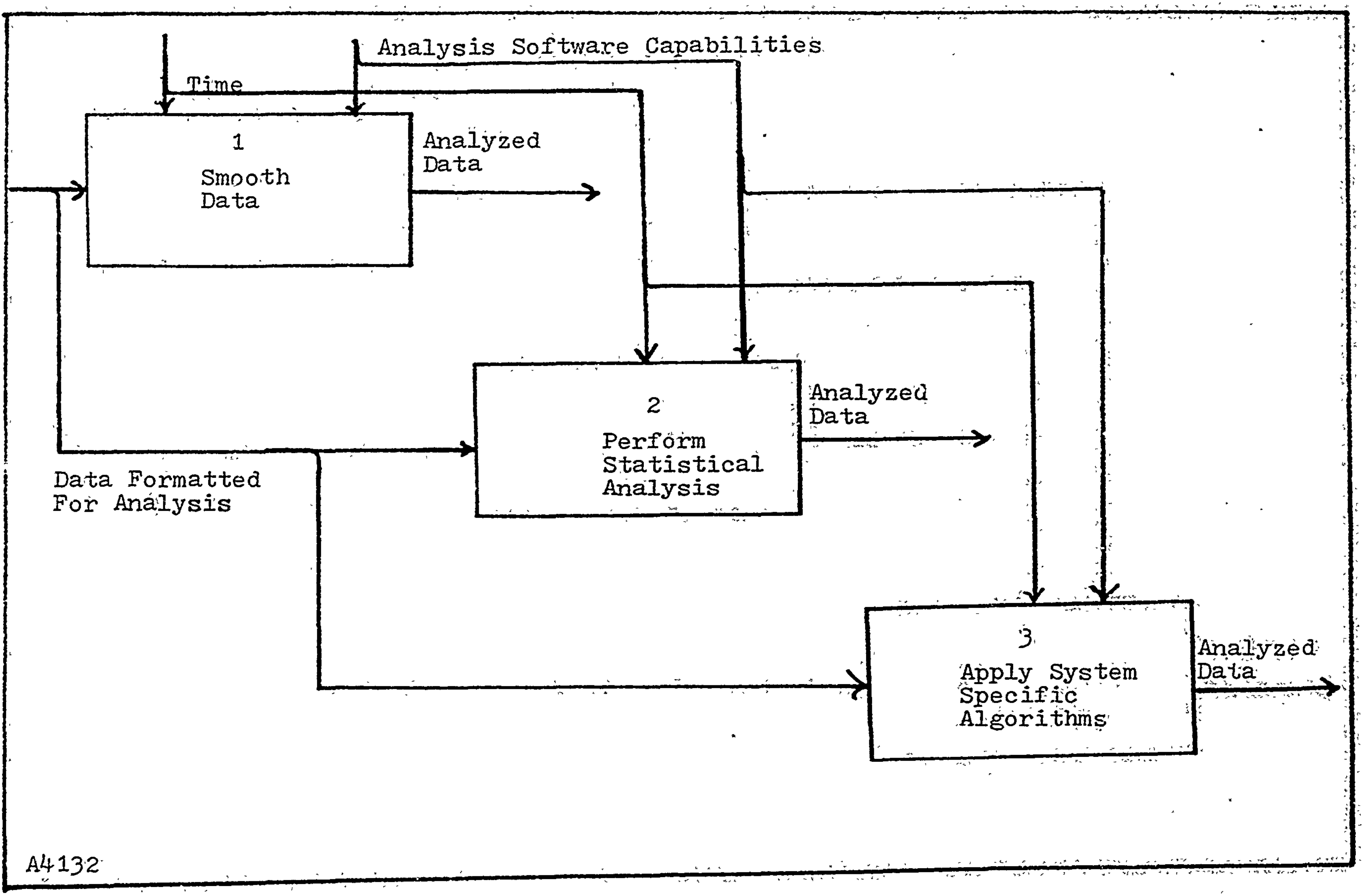
Test Software



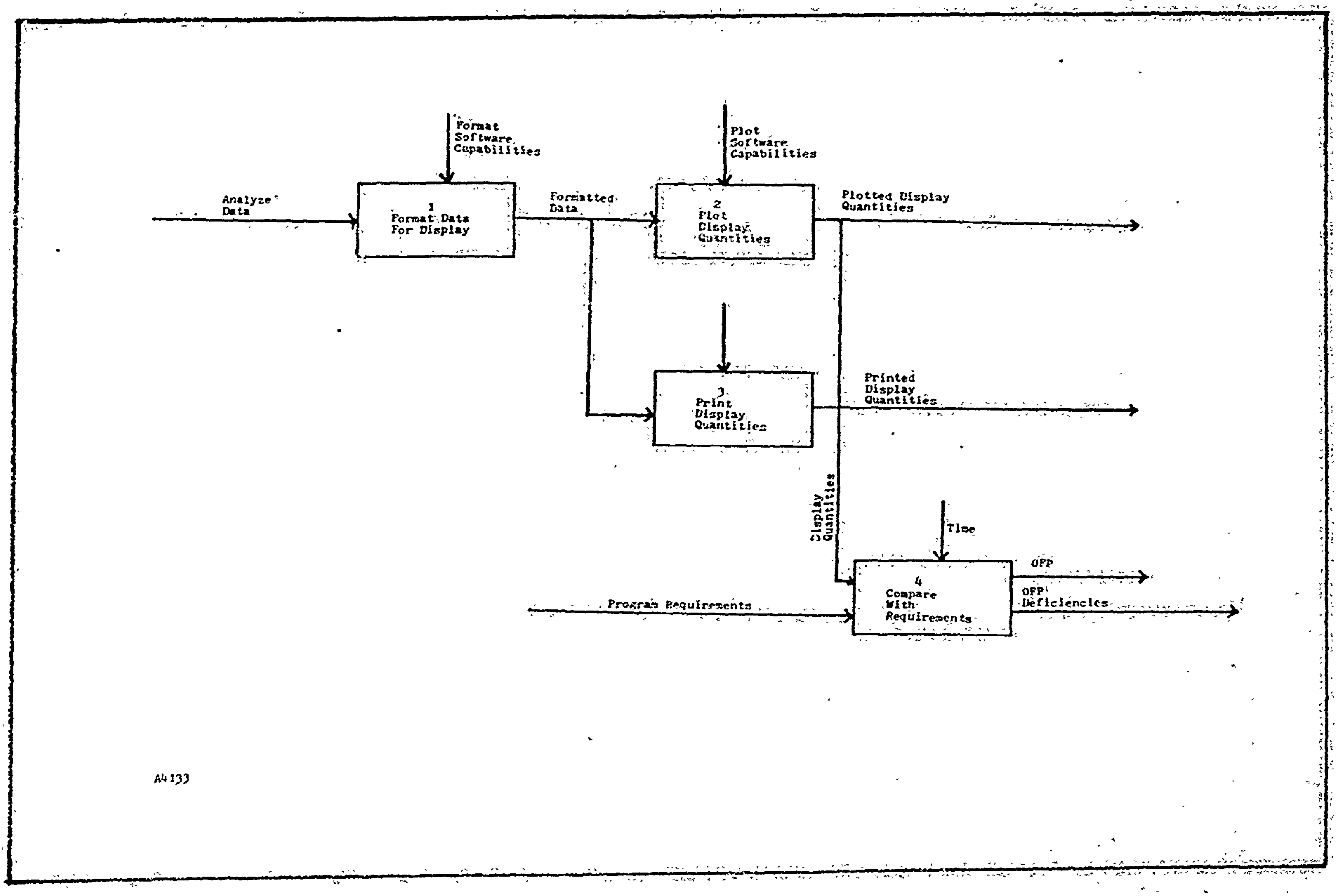
Analyze Program Test Results


Raw
Test
Data
1
Unpack
Data
Tapes
Test
Data
Edit Capabilities
Time
2
Edit
Data
Edited Test Data
Data
Formatter
Capabilities
3
Format
Data
Formatted
Test Data
A4131

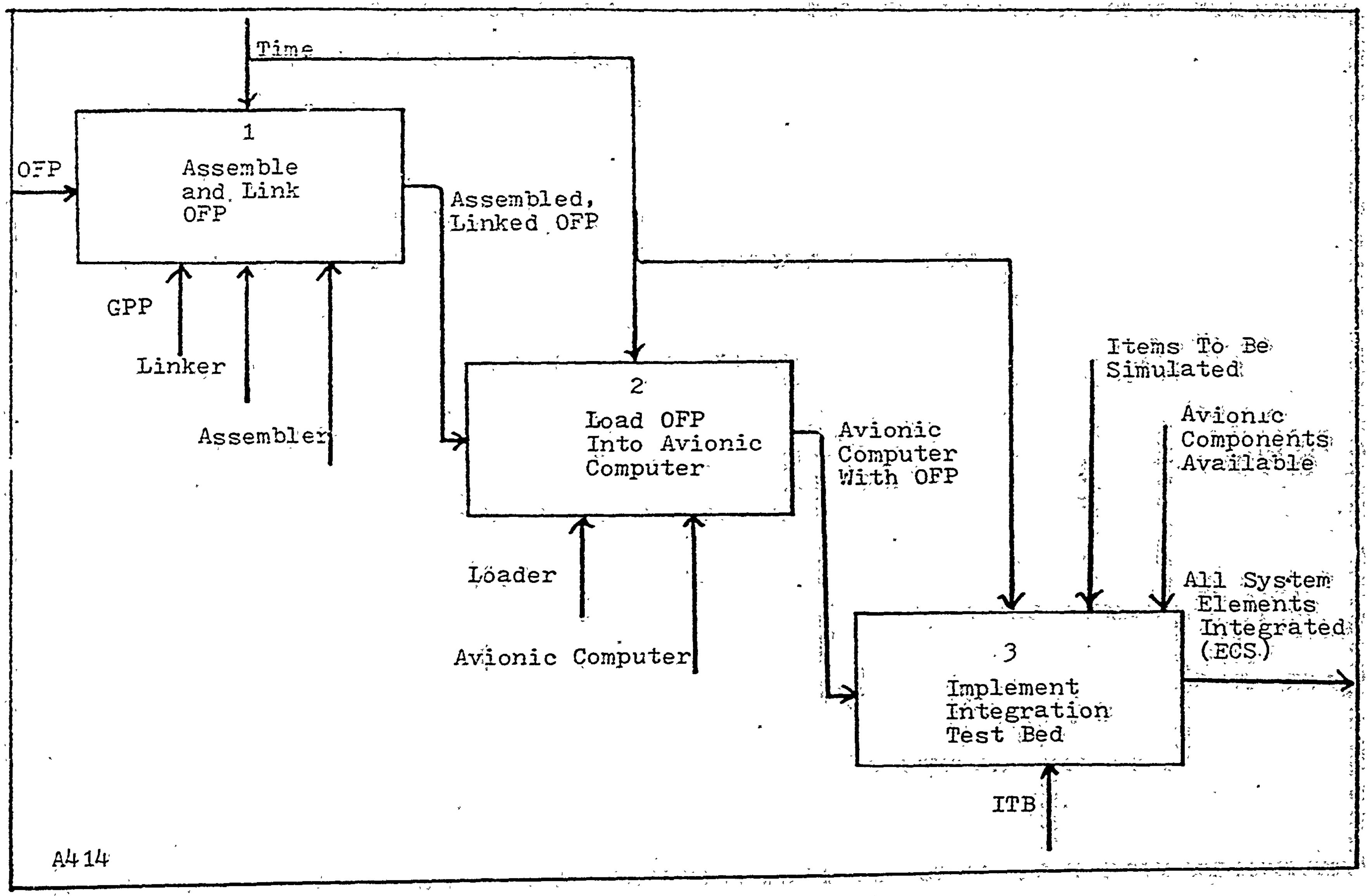
Reduce Data


Analysis Software Capabilities
Time
1
Smooth Data
Analyzed Data
2
Perform Statistical Analysis
Analyzed Data
Data Formatted For Analysis
3
Apply System Specific Algorithms
Analyzed Data
A4132

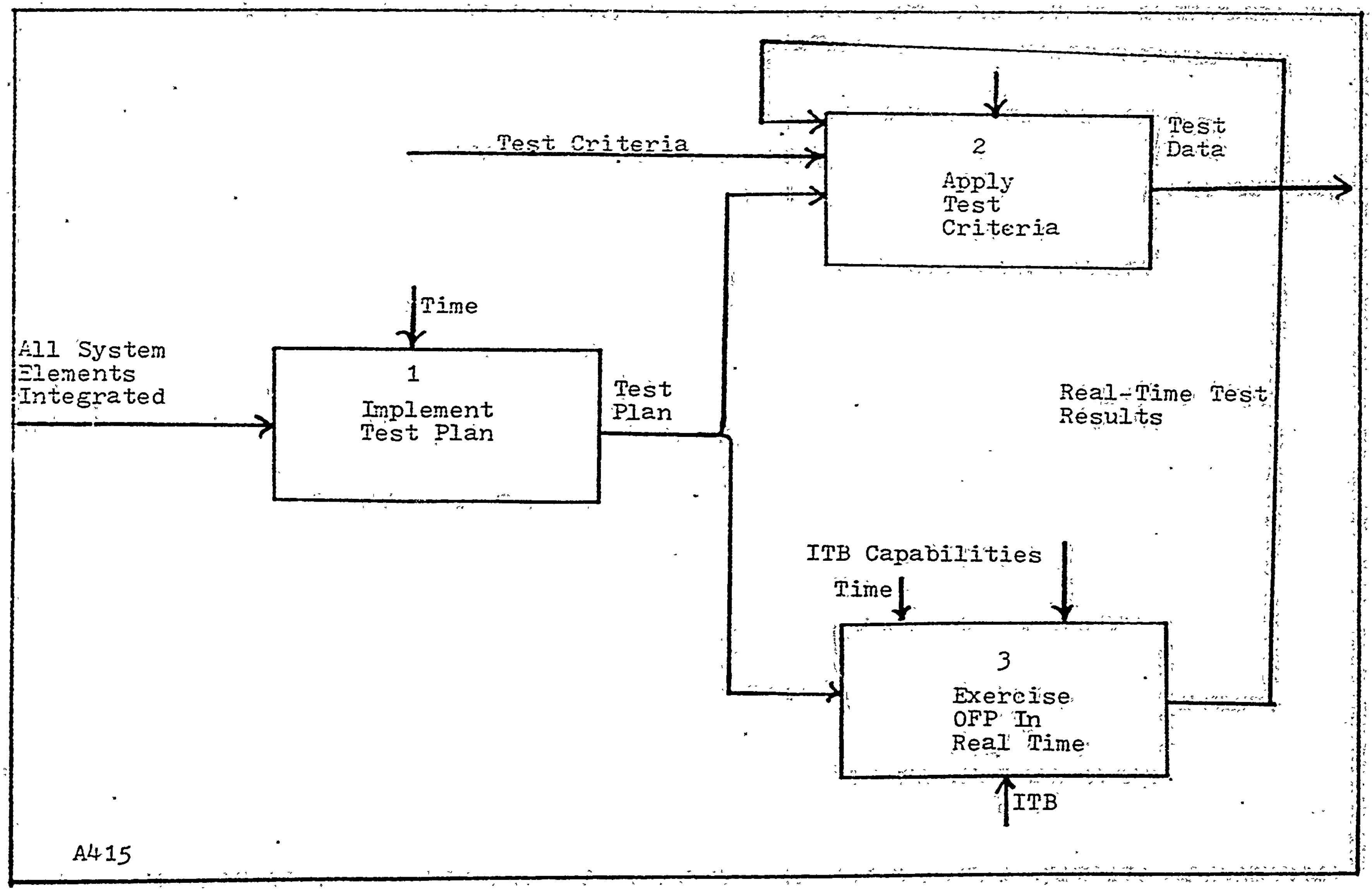
Analyze Data


Analyze Data
1 Format Data For Display
Format Software Capabilities
Formatted Data
2 Plot Display Quantities
Plot Software Capabilities
Plotted Display Quantities
3 Print Display Quantities
Printed Display Quantities
Display Quantities
Time
4 Compare With Requirements
Program Requirements
OFP
OFP Deficiencies
A4133

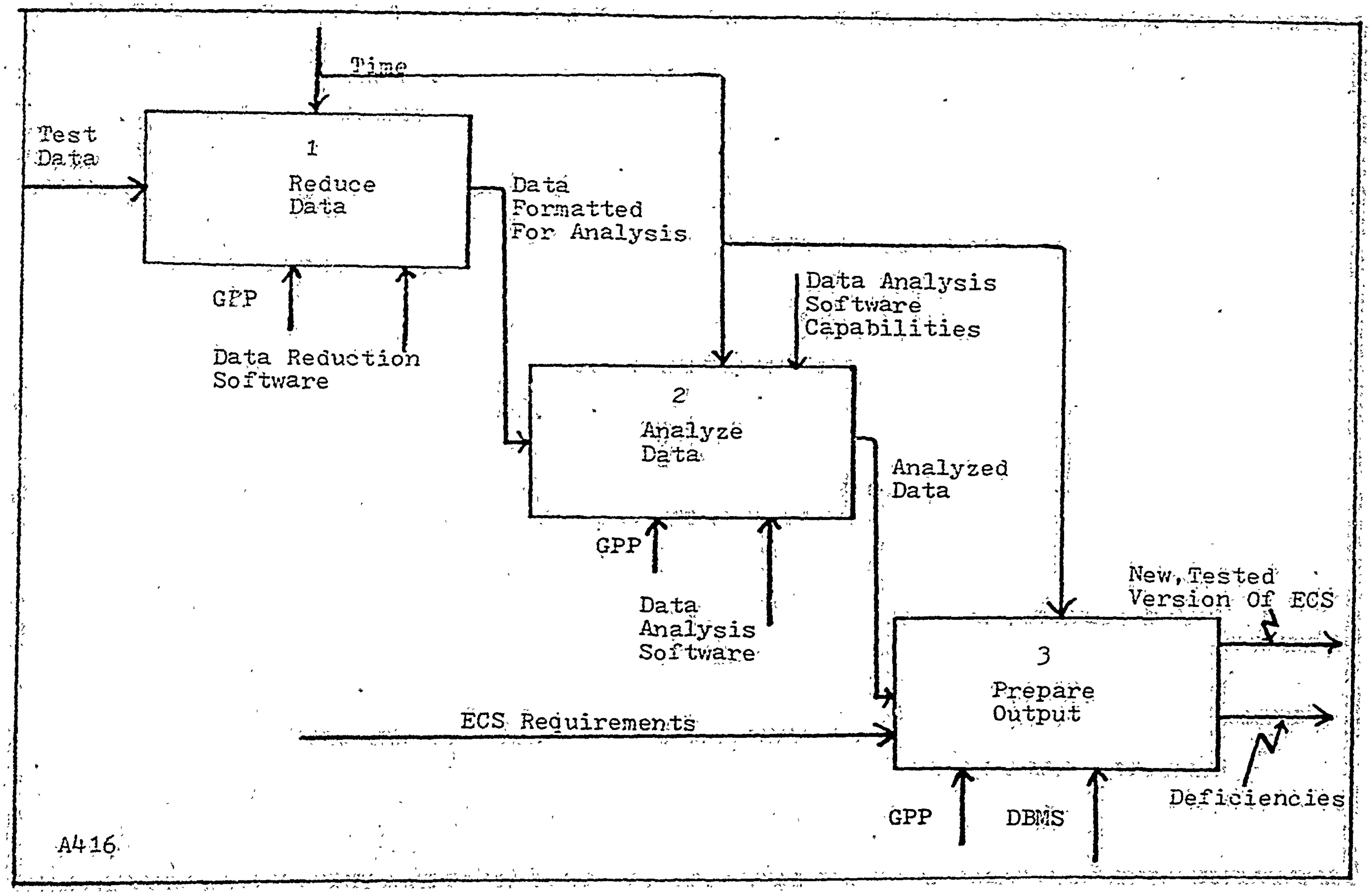
Prepare Output


Time
1
Assemble and Link OFP
OFP
Assembled, Linked OFP
GPP
Linker
Assembler
2
Load OFP Into Avionic Computer
Loader
Avionic Computer
Avionic Computer With OFP
Items To Be Simulated
Avionic Components Available
All System Elements Integrated (ECS)
3
Implement Integration Test Bed
ITB
A414

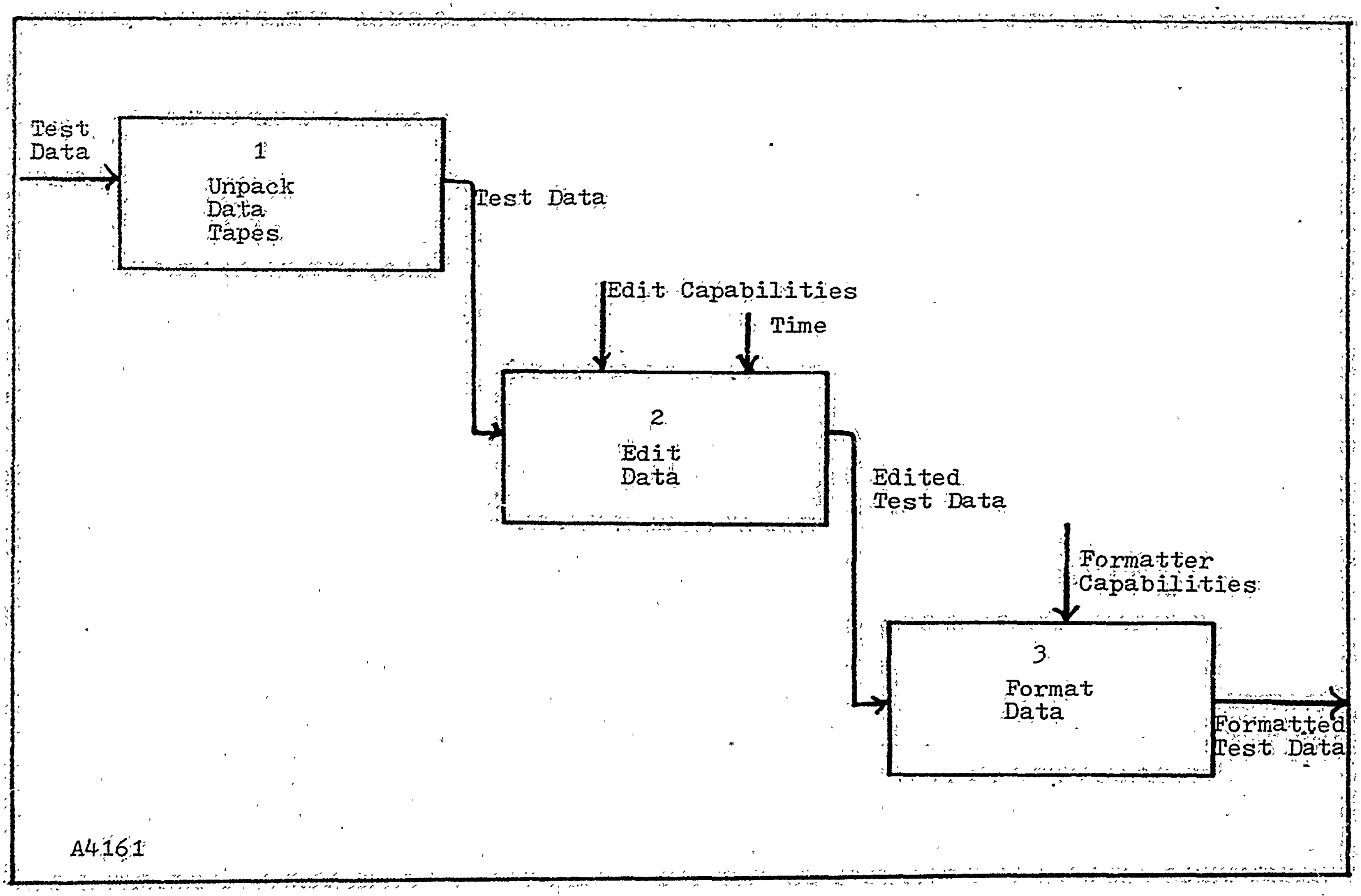
Integrate System


Test Criteria
2
Apply
Test
Criteria
Test
Data
Time
All System
Elements
Integrated
1
Implement
Test Plan
Test
Plan
Real-Time Test
Results
ITB Capabilities
Time
3
Exercise
OFP In
Real Time
ITB
A415

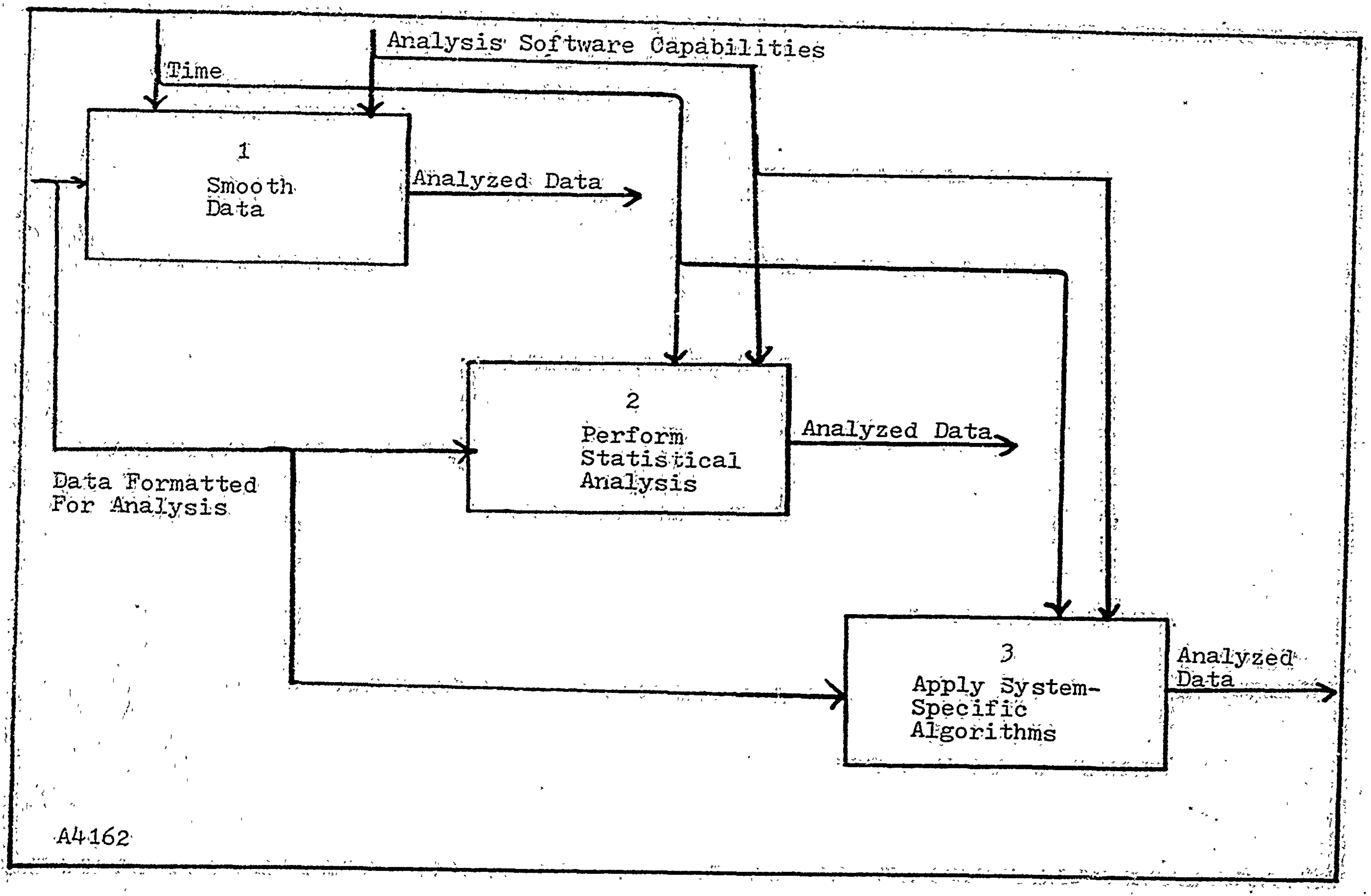
Test System



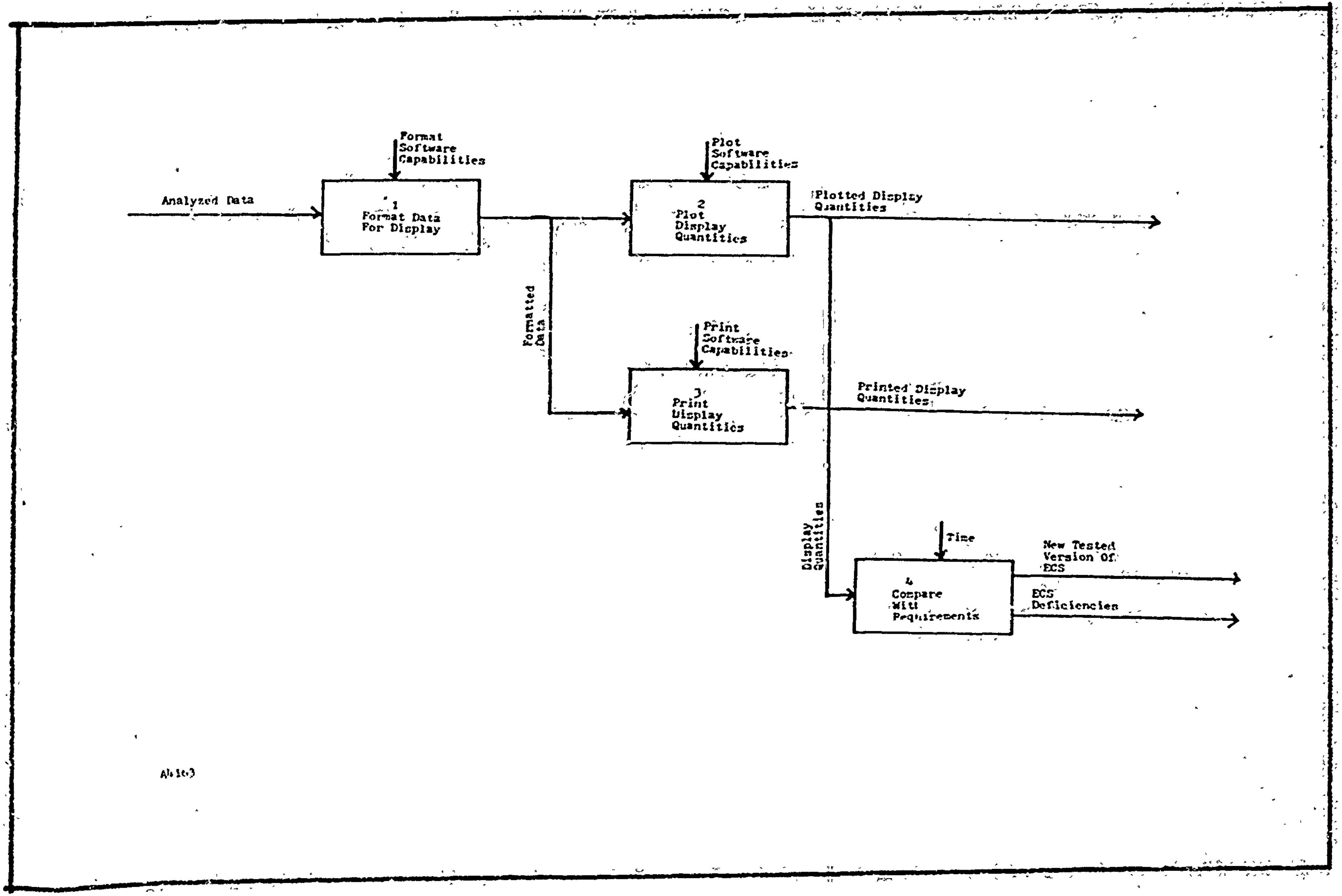
Analyze ECS Test Results


Test Data
1
Unpack Data Tapes
Test Data
Edit Capabilities
Time
2
Edit Data
Edited Test Data
Formatter Capabilities
3
Format Data
Formatted Test Data
A4161

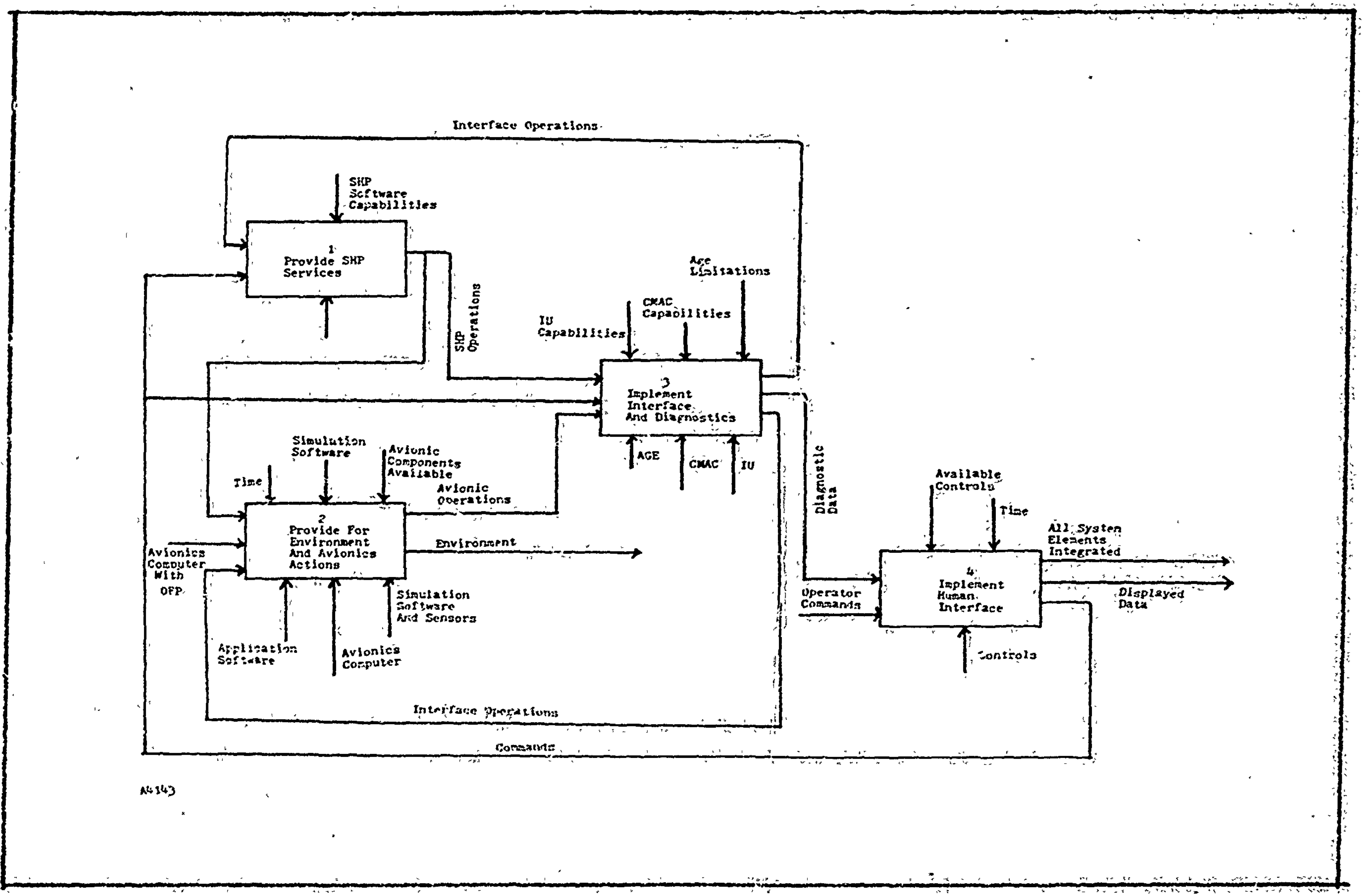
Reduce Data


Time
Analysis Software Capabilities
1
Smooth Data
Analyzed Data
2
Perform Statistical Analysis
Analyzed Data
Data Formatted For Analysis
3
Apply System-Specific Algorithms
Analyzed Data
A4162

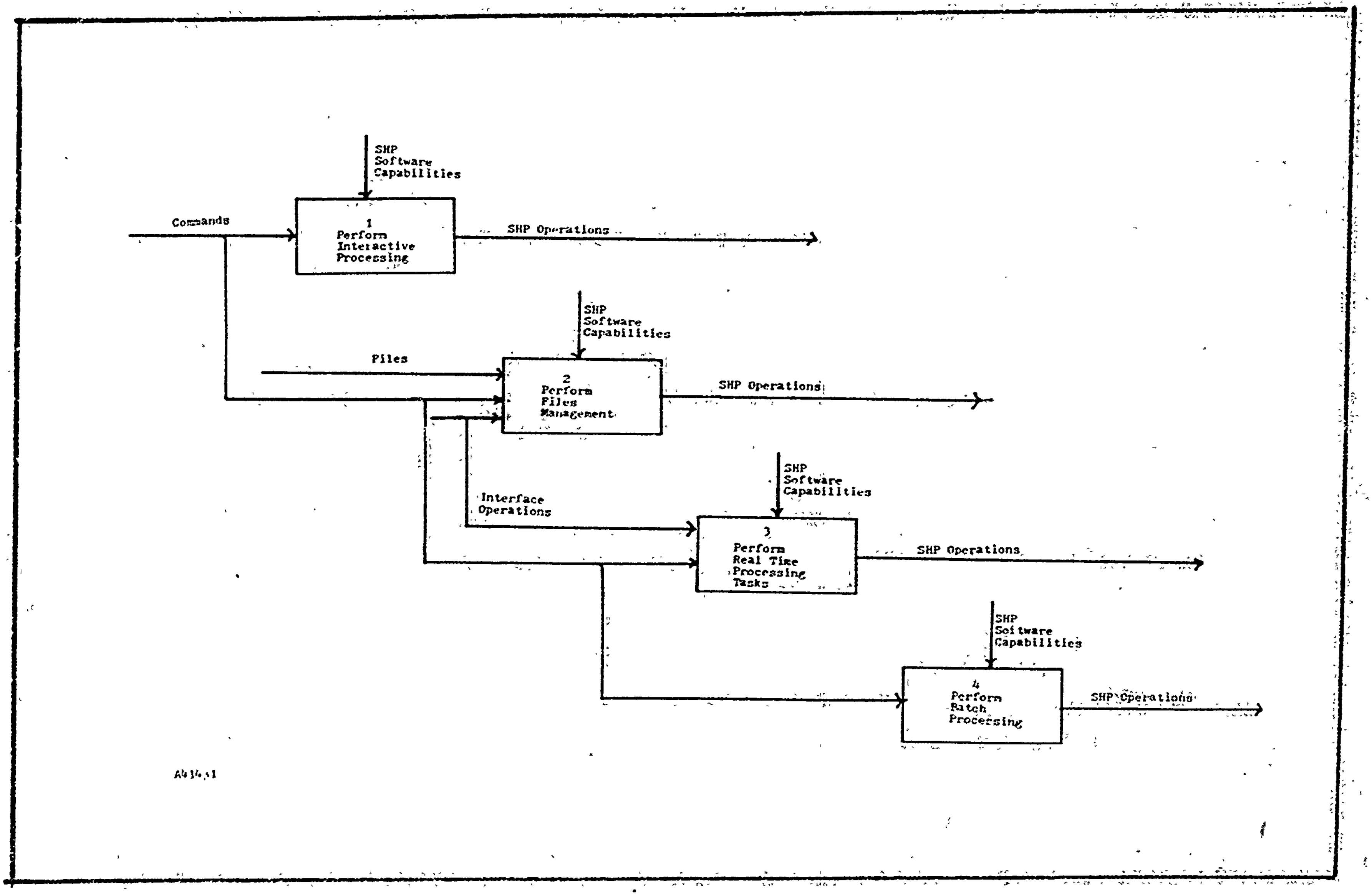
Analyze Data


Analyzed Data
Format Software Capabilities
1 Format Data For Display
Formatted Data
Plot Software Capabilities
2 Plot Display Quantities
Plotted Display Quantities
Print Software Capabilities
3 Print Display Quantities
Printed Display Quantities
Display Quantities
Time
4 Compare With Requirements
New Tested Version Of ECS
ECS Deficiencies
A4163

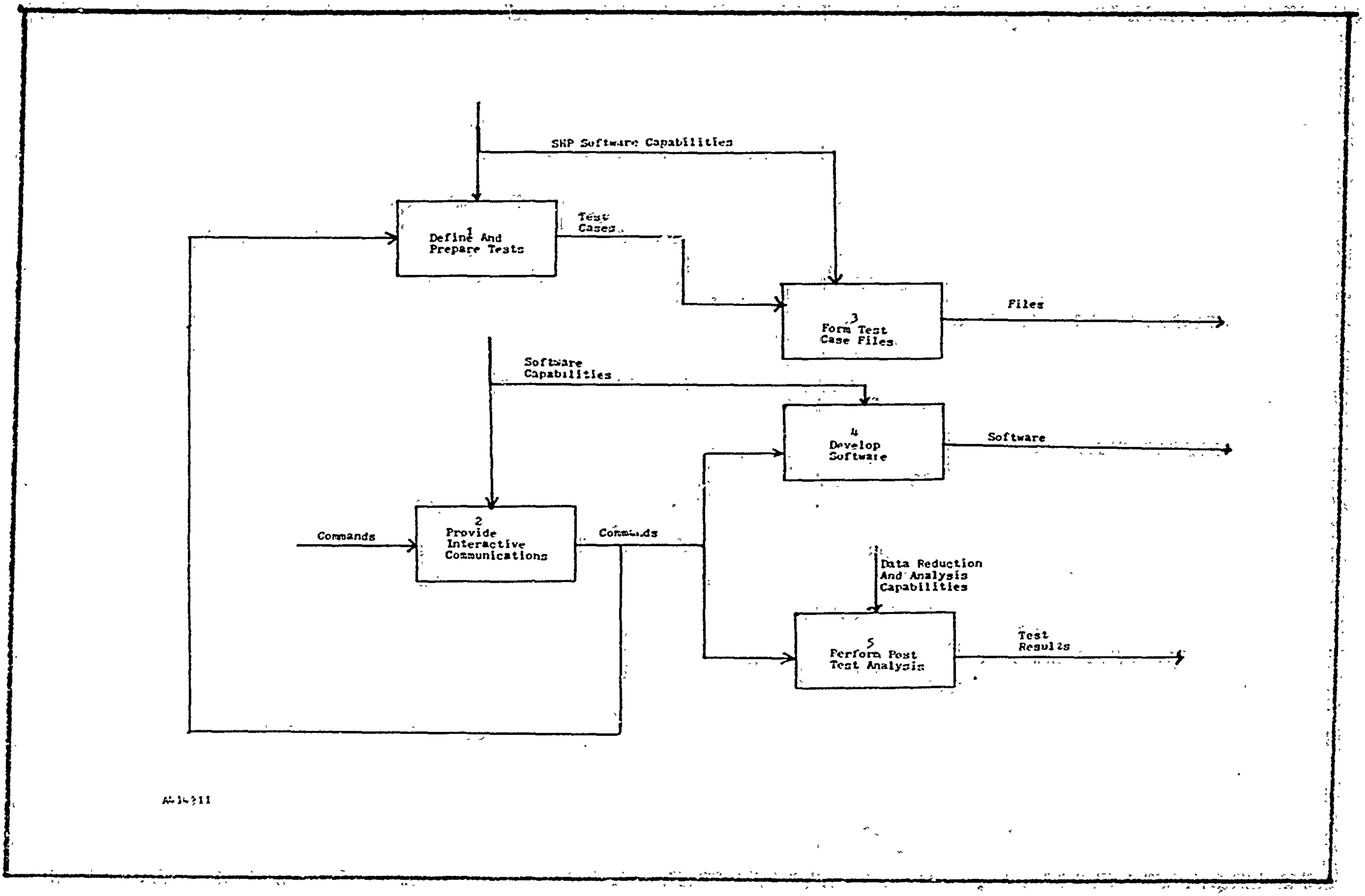
Prepare Output



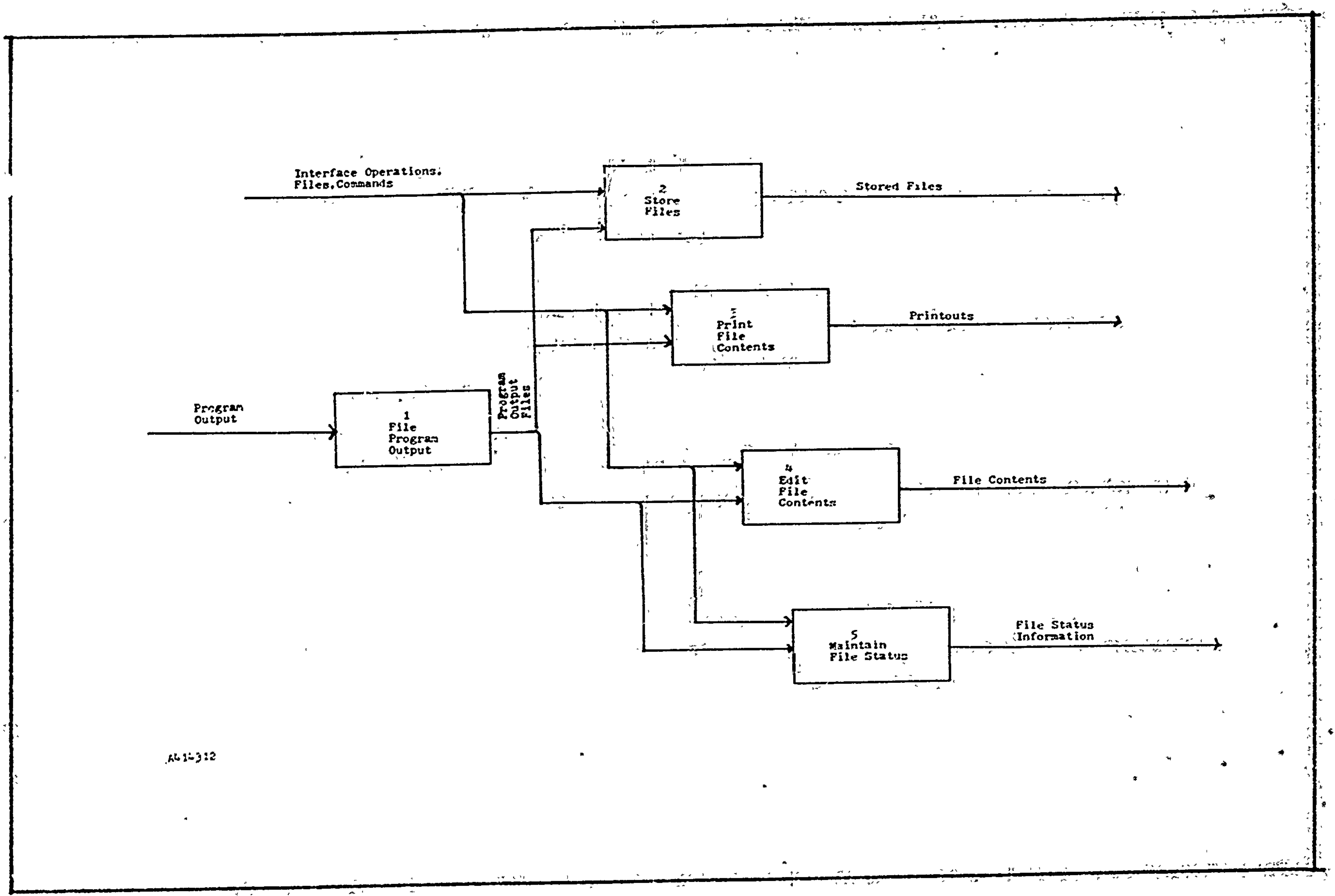
Implement Integration Test Bed


SHP Software Capabilities
Commands
1 Perform Interactive Processing
SHP Operations
SHP Software Capabilities
Piles
2 Perform Piles Management
SHP Operations
SHP Software Capabilities
Interface Operations
3 Perform Real Time Processing Tasks
SHP Operations
SHP Software Capabilities
4 Perform Batch Processing
SHP Operations
A414.1

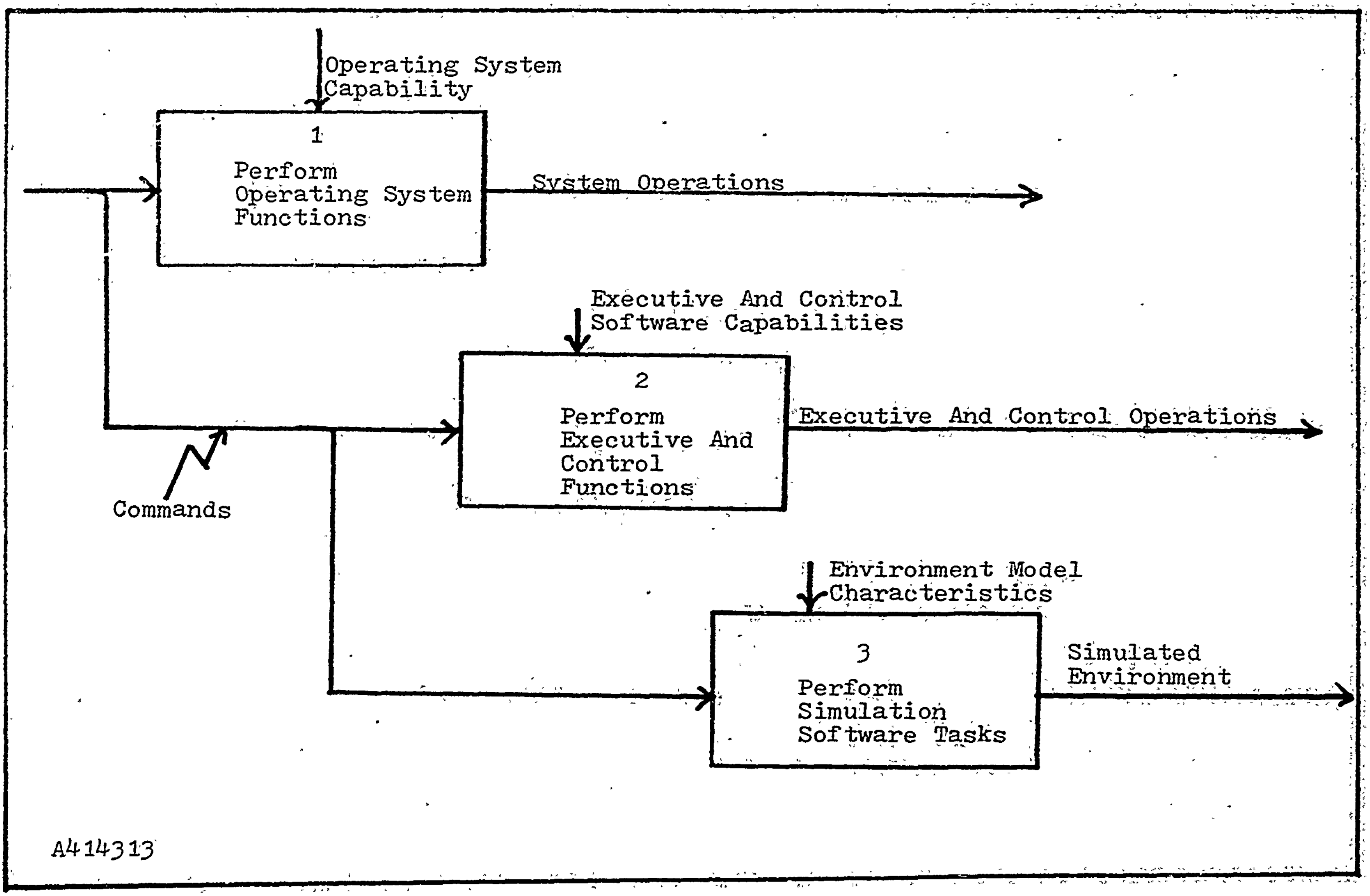
Provide SHP Services


SHP Software Capabilities
1 Define And Prepare Tests
Test Cases
3 Form Test Case Files
Files
Software Capabilities
4 Develop Software
Software
Commands
2 Provide Interactive Communications
Commands
Data Reduction And Analysis Capabilities
5 Perform Post Test Analysis
Test Results
A-14311

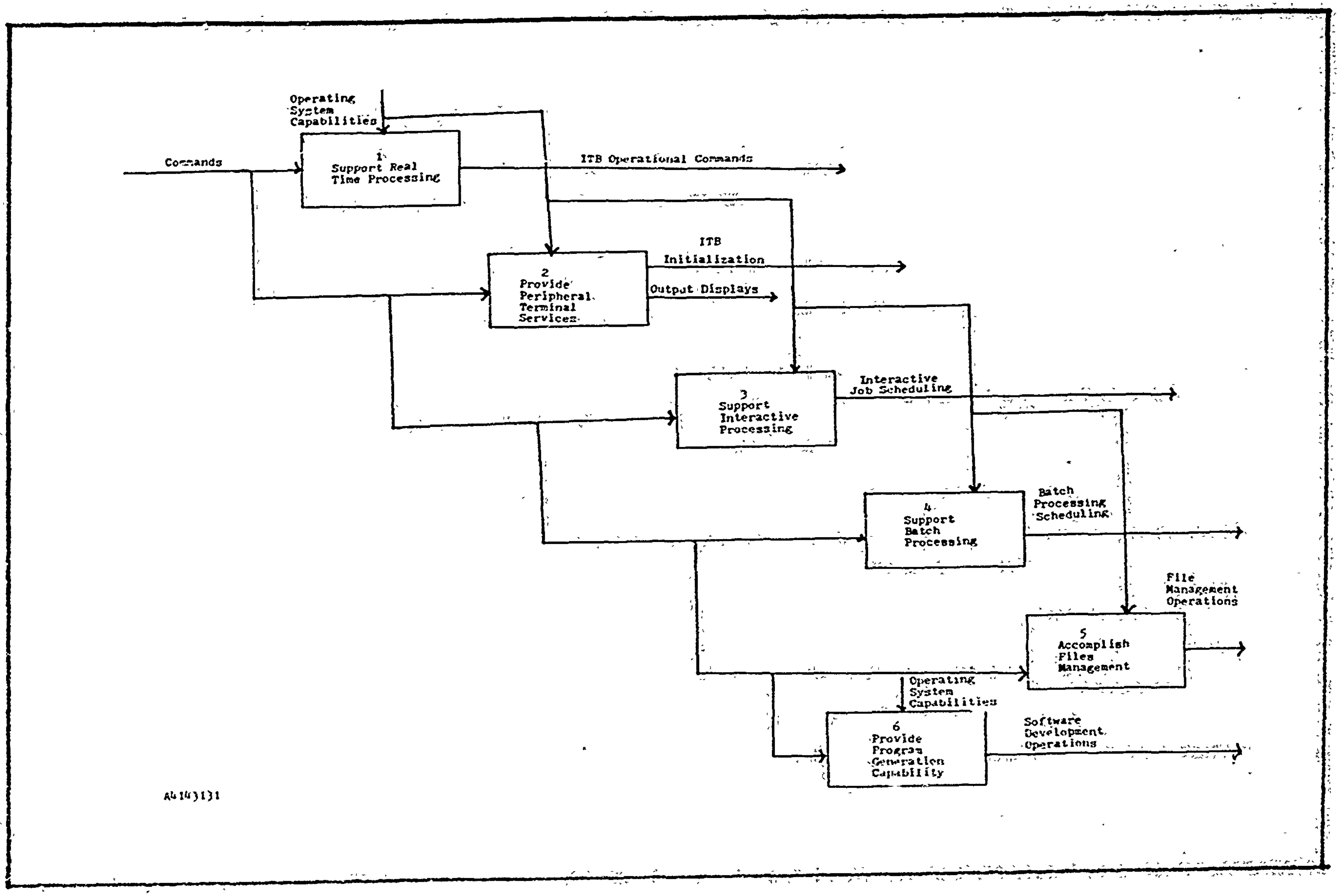
Perform Interactive Processing



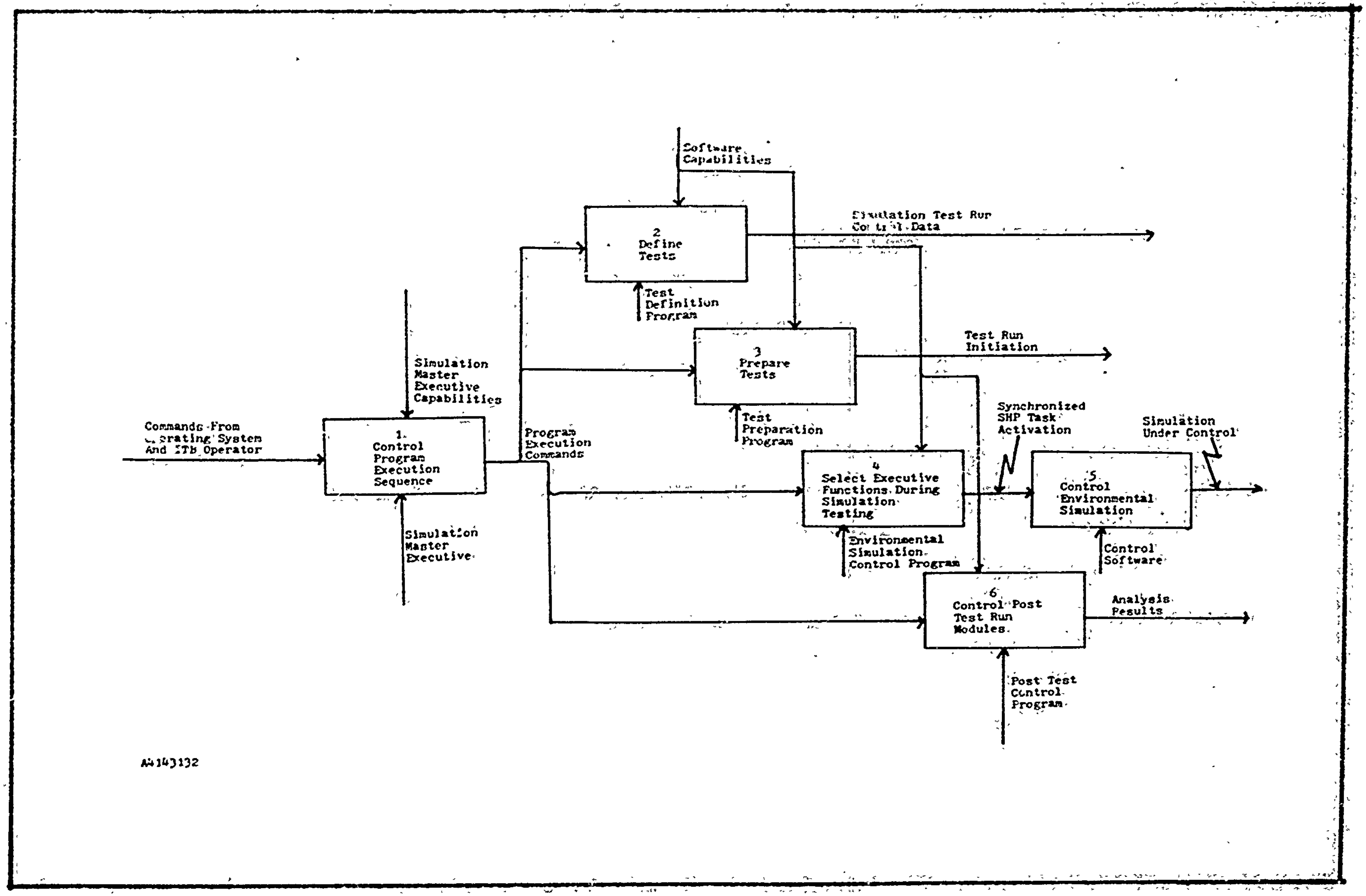
Perform Files Management



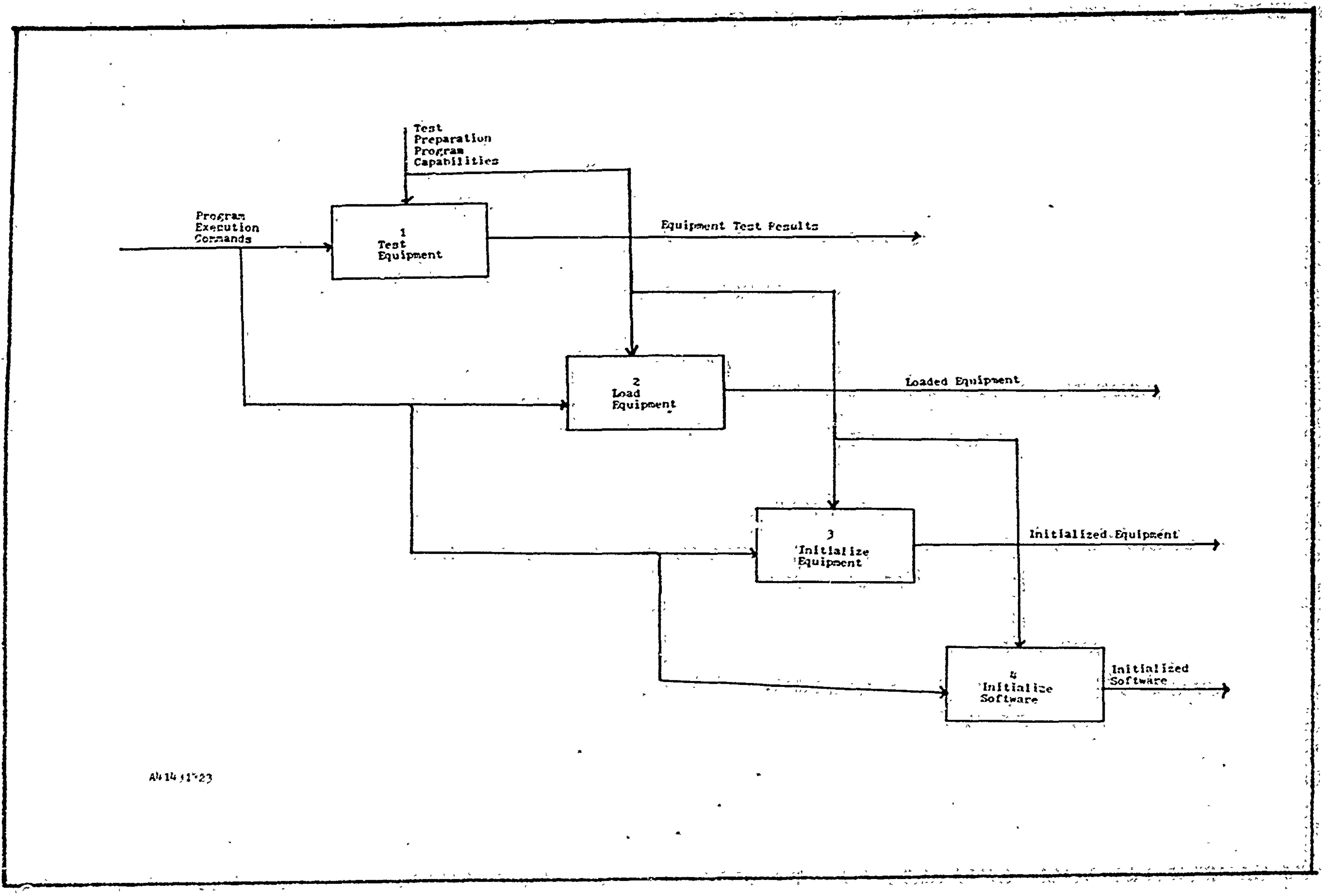
Perform Real-Time Processing Tasks


Operating System Capabilities
Commands
1 Support Real Time Processing
ITB Operational Commands
2 Provide Peripheral Terminal Services
ITB Initialization
Output Displays
3 Support Interactive Processing
Interactive Job Scheduling
4 Support Batch Processing
Batch Processing Scheduling
File Management Operations
5 Accomplish Files Management
Operating System Capabilities
6 Provide Program Generation Capability
Software Development Operations
A4143131

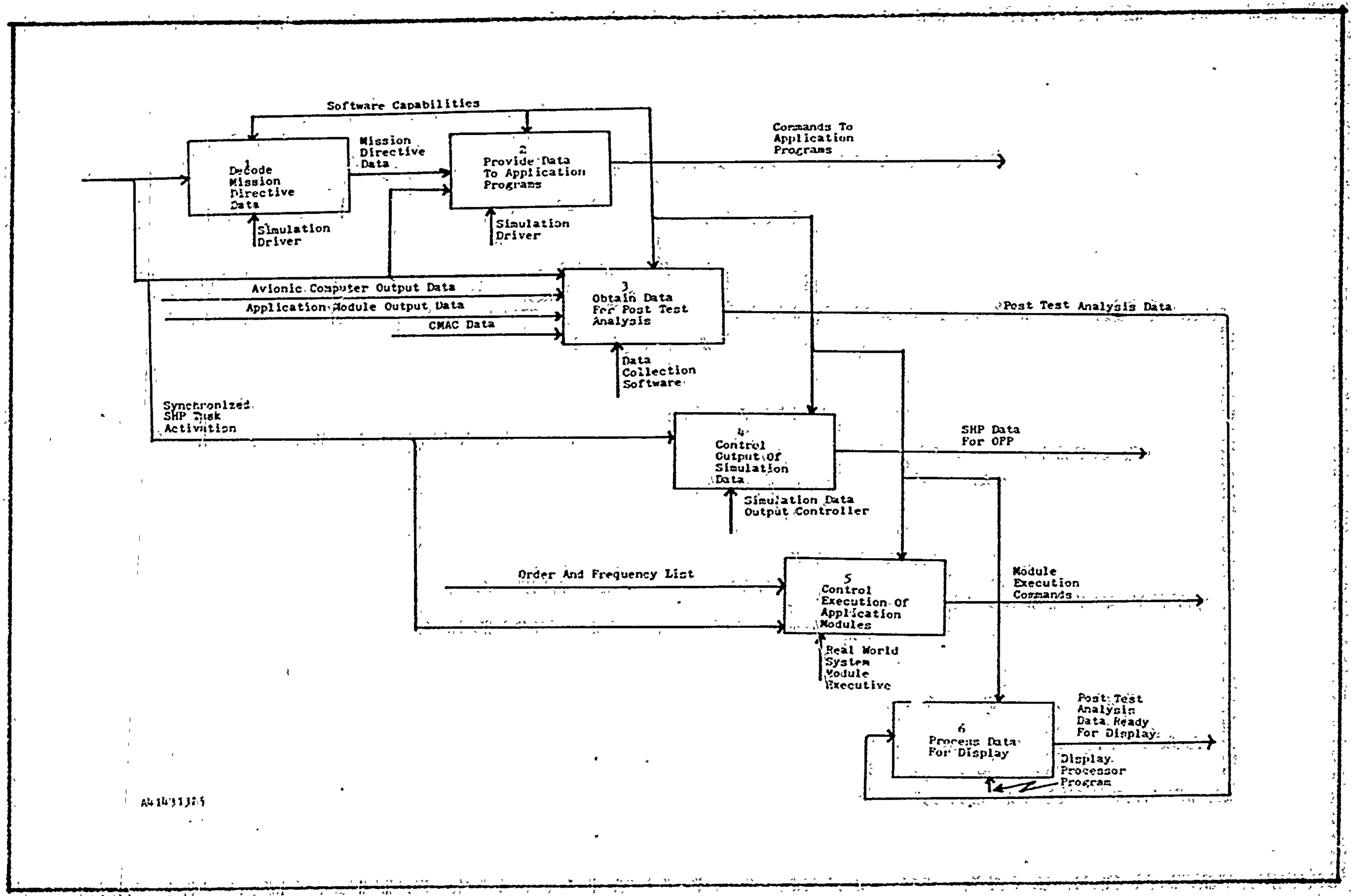
Perform Operating System Functions


Software Capabilities
2 Define Tests
Test Definition Program
Simulation Test Rur Control Data
3 Prepare Tests
Test Run Initiation
Test Preparation Program
Simulation Master Executive Capabilities
Commands From Operating System And ITB Operator
1. Control Program Execution Sequence
Program Execution Commands
Simulation Master Executive
Synchronized SHP Task Activation
Simulation Under Control
4 Select Executive Functions During Simulation Testing
5 Control Environmental Simulation
Environmental Simulation Control Program
Control Software
6 Control Post Test Run Modules
Analysis Pesults
Post Test Control Program
A4143132

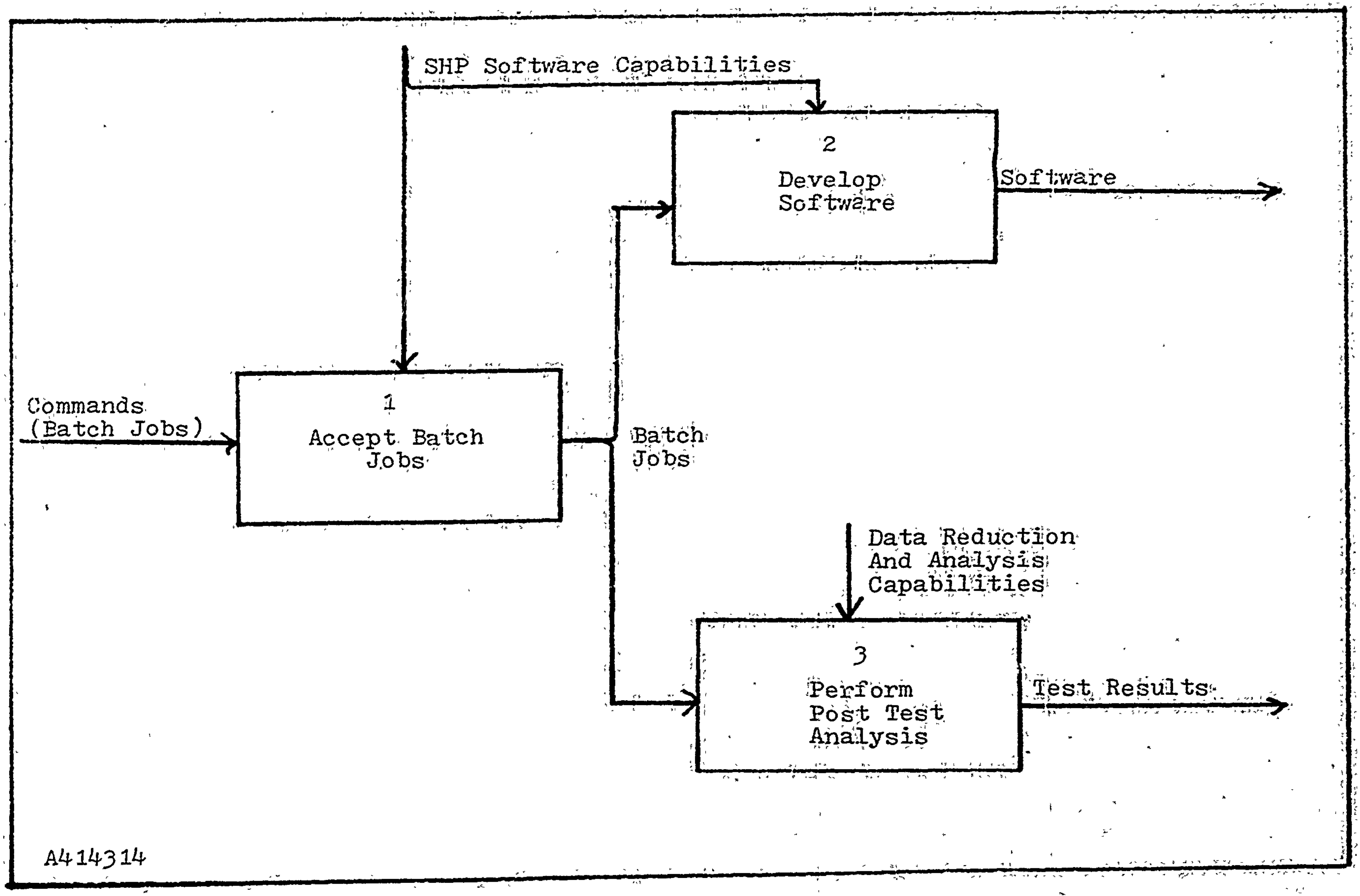
Perform Executive And Control Functions



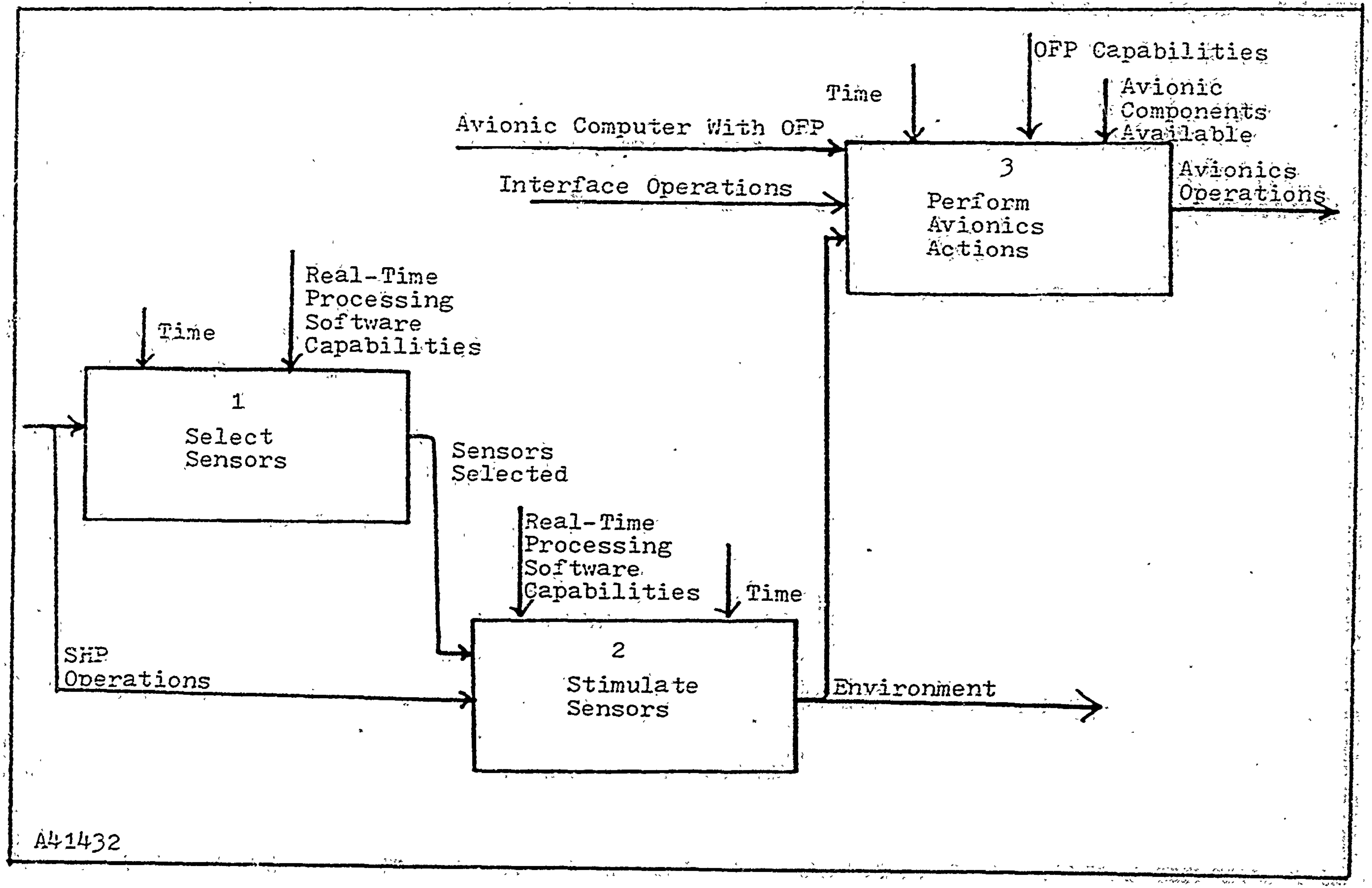
Prepare Tests



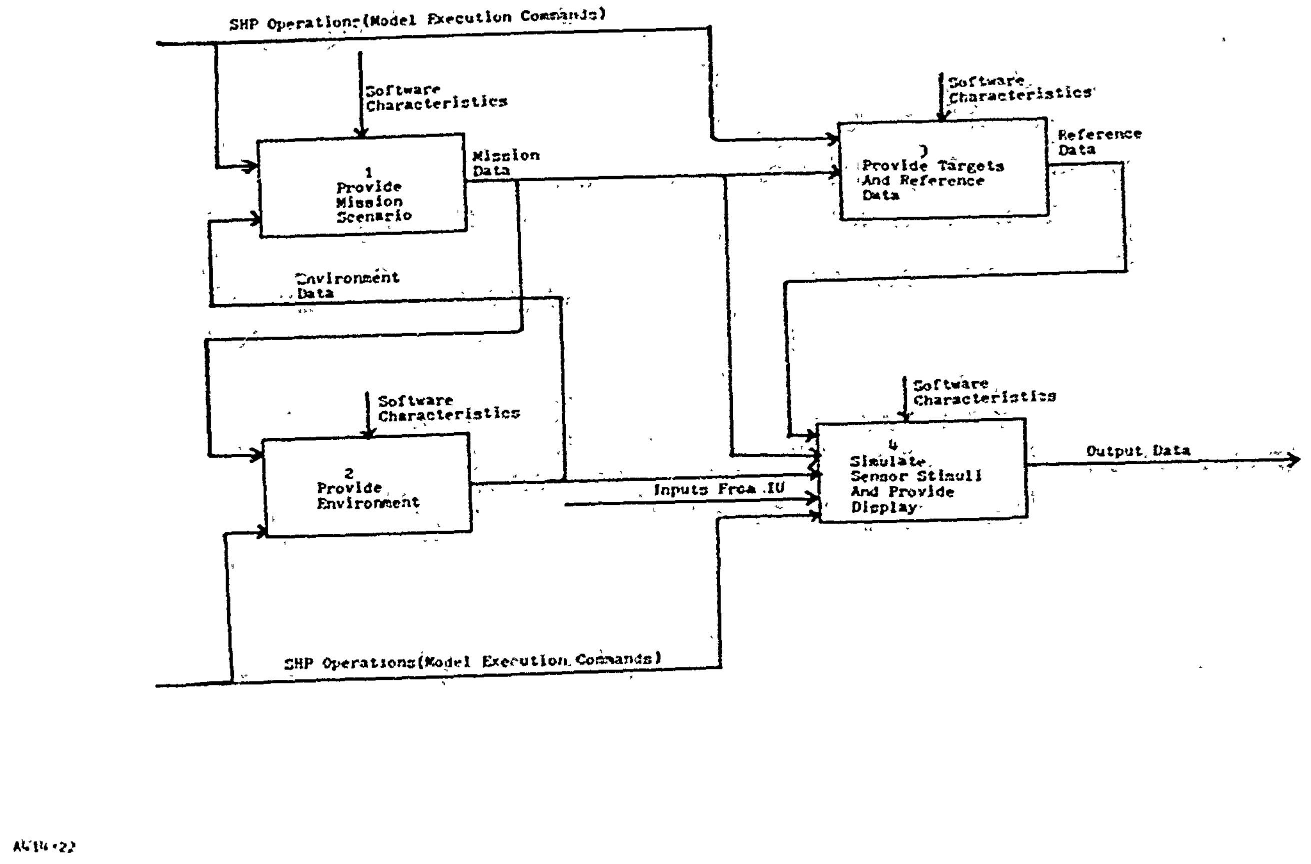
Control Environmental Simulation



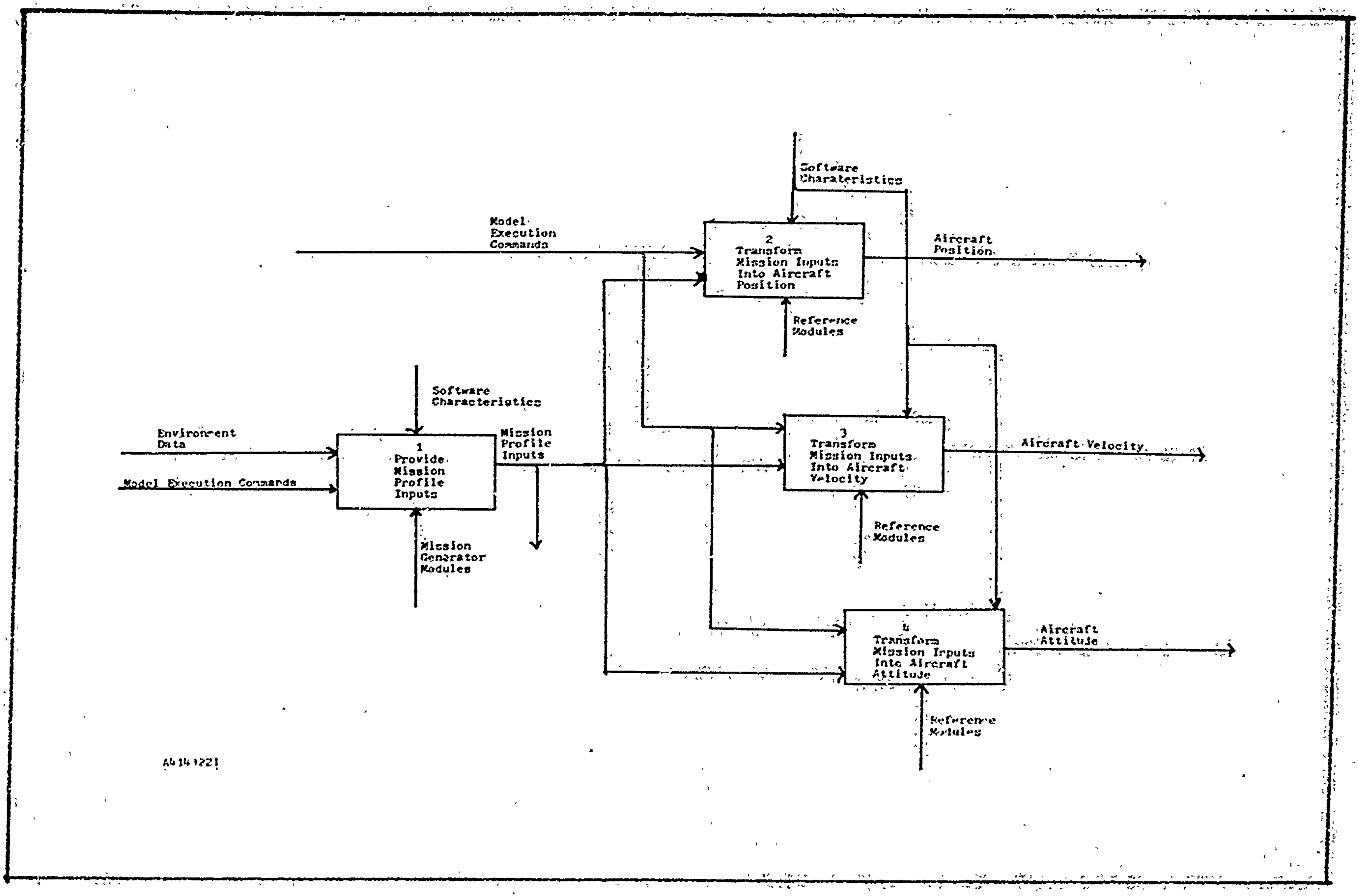
Perform Batch Processing



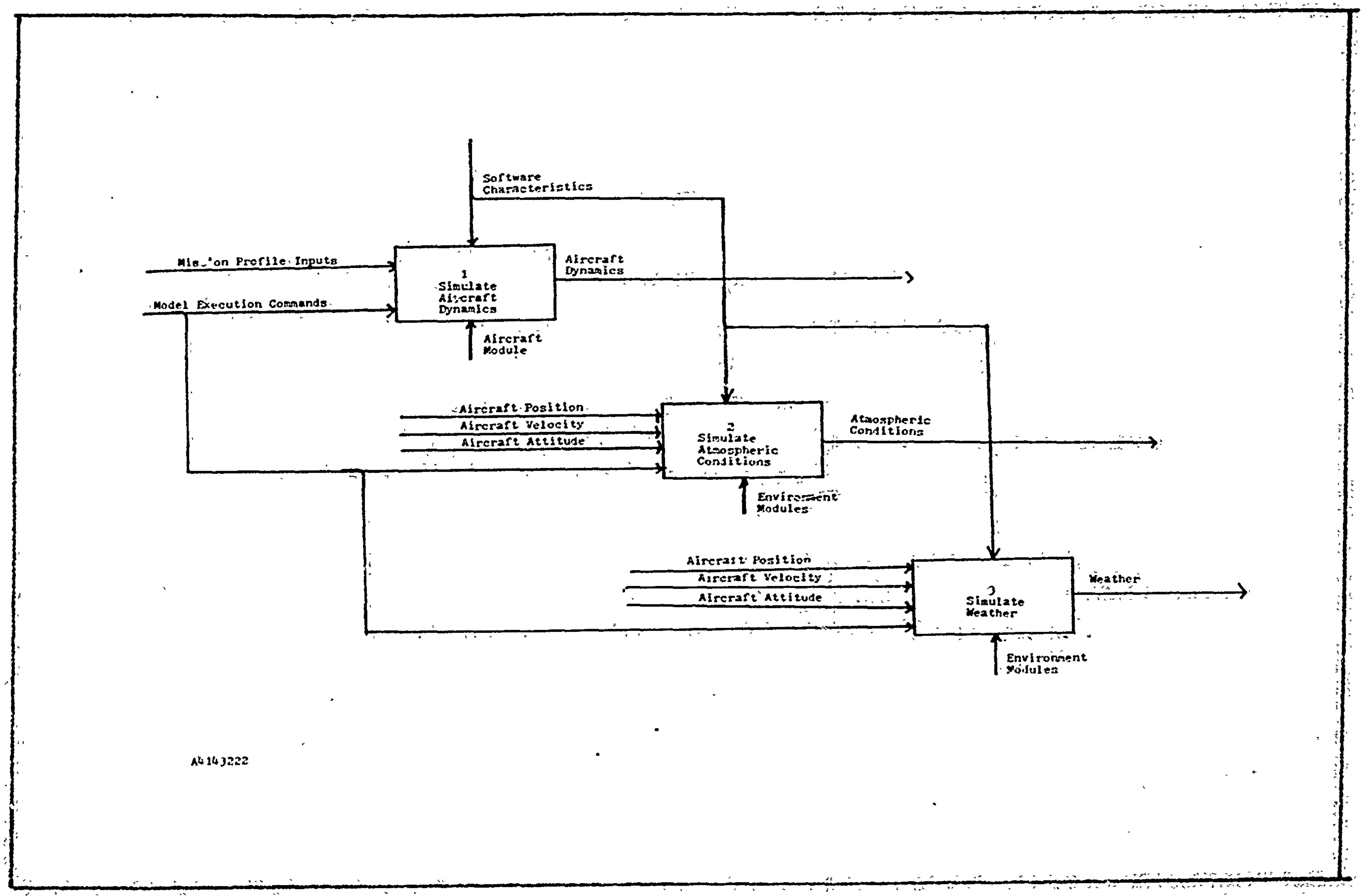
Provide For Environment And Avionics Actions



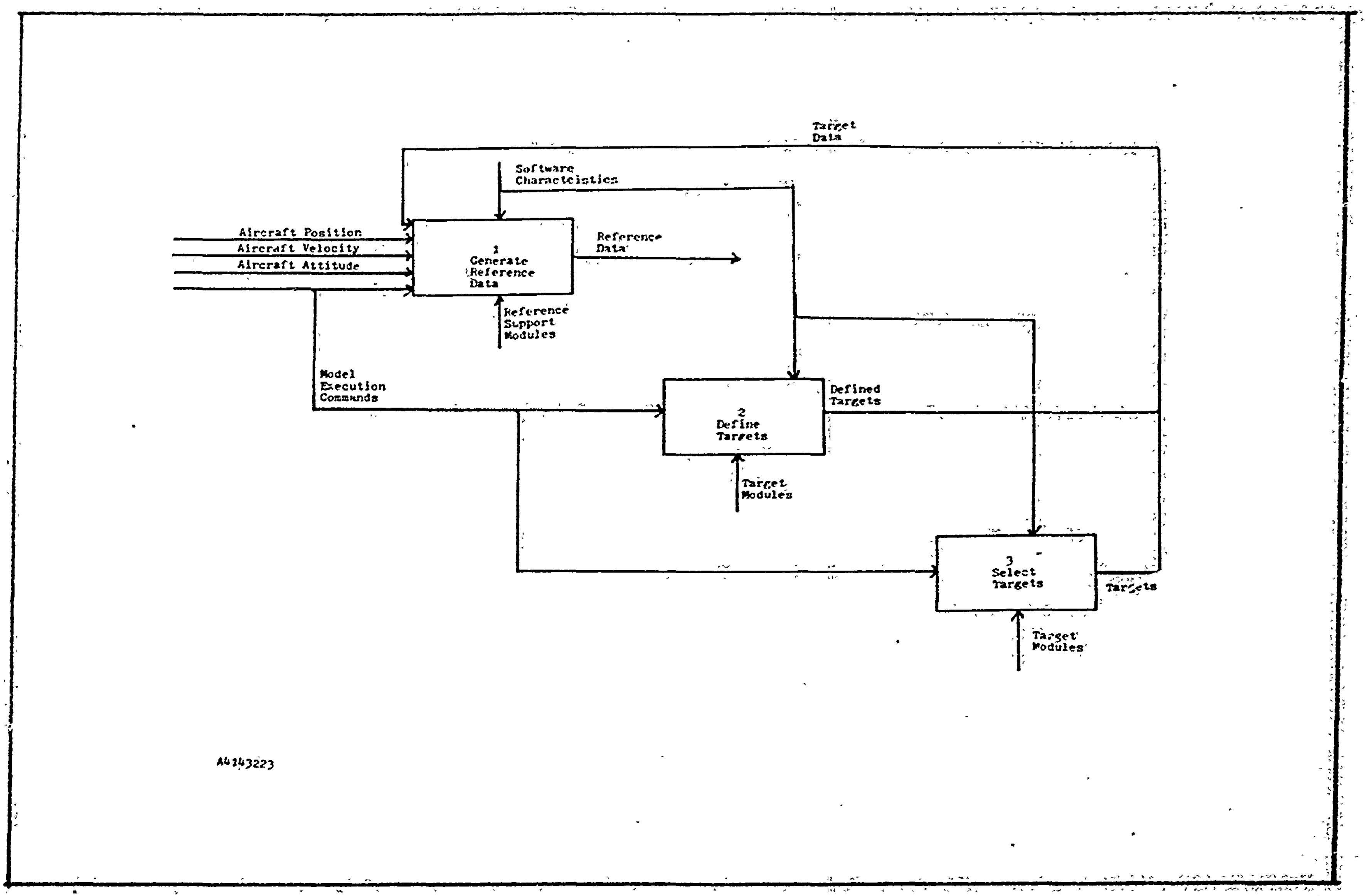
Stimulate Sensors (Function Of ITB Application Modules)



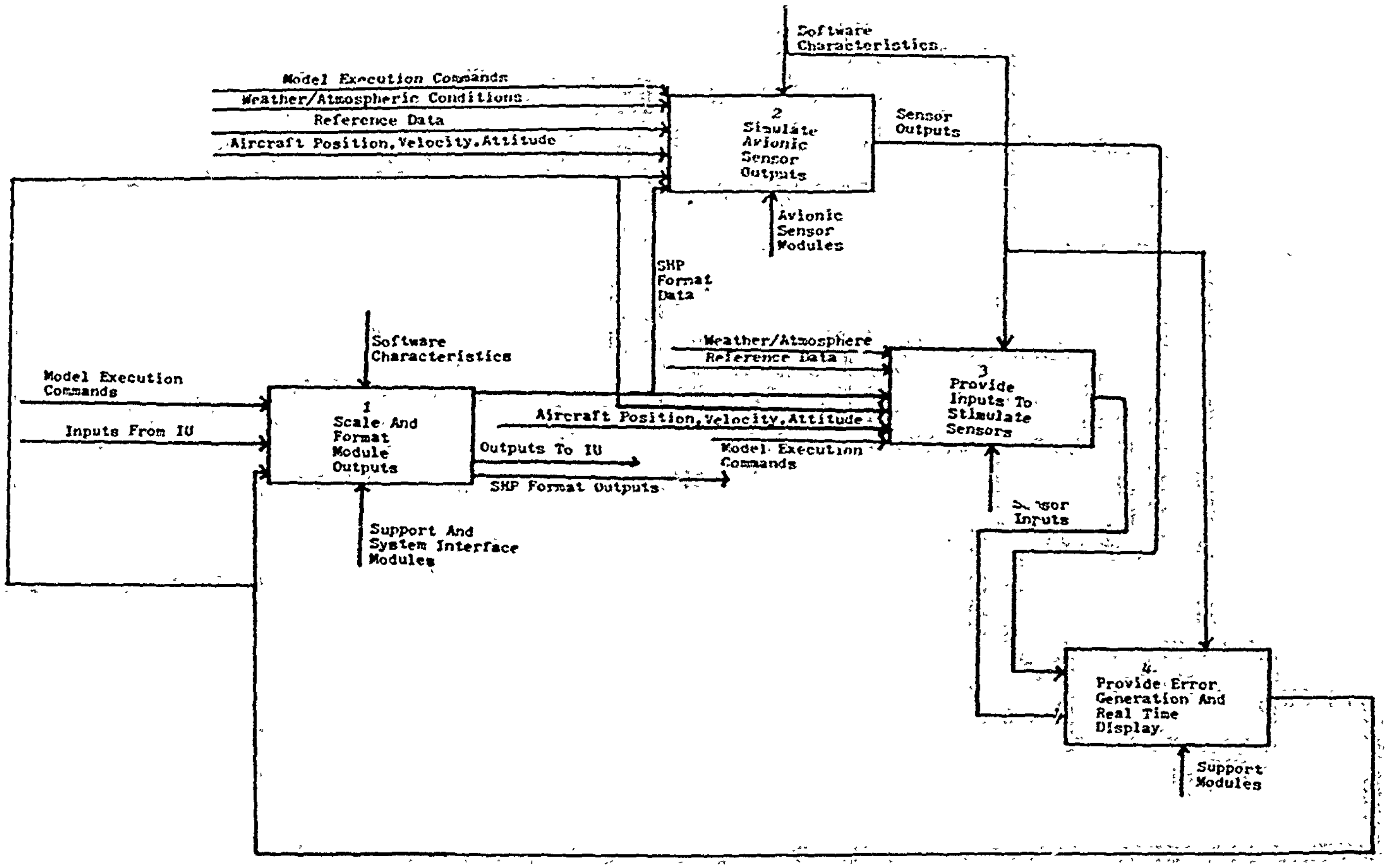
Provide Mission Scenario



Provide Environment

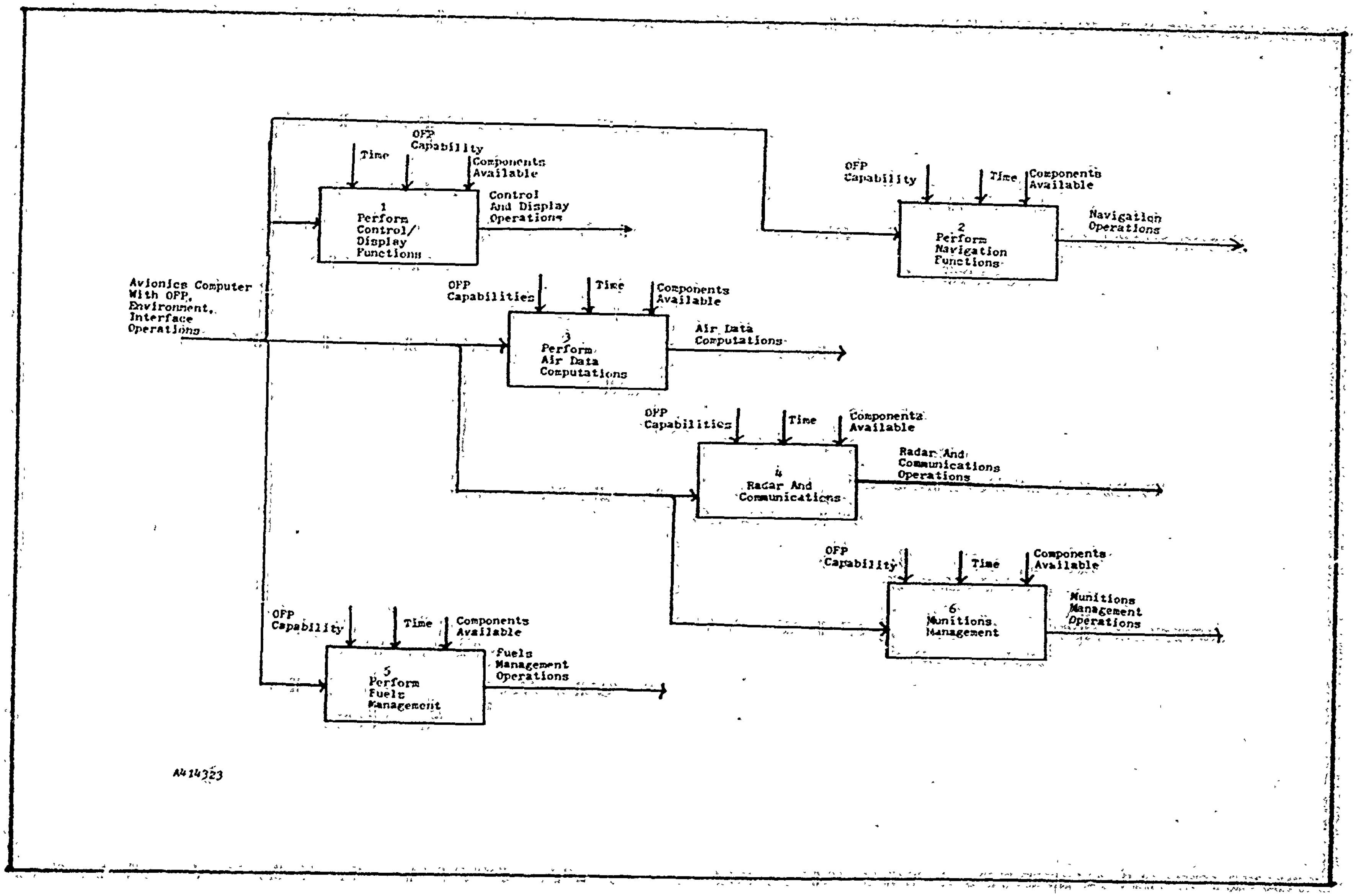


Provide Target And Reference Data

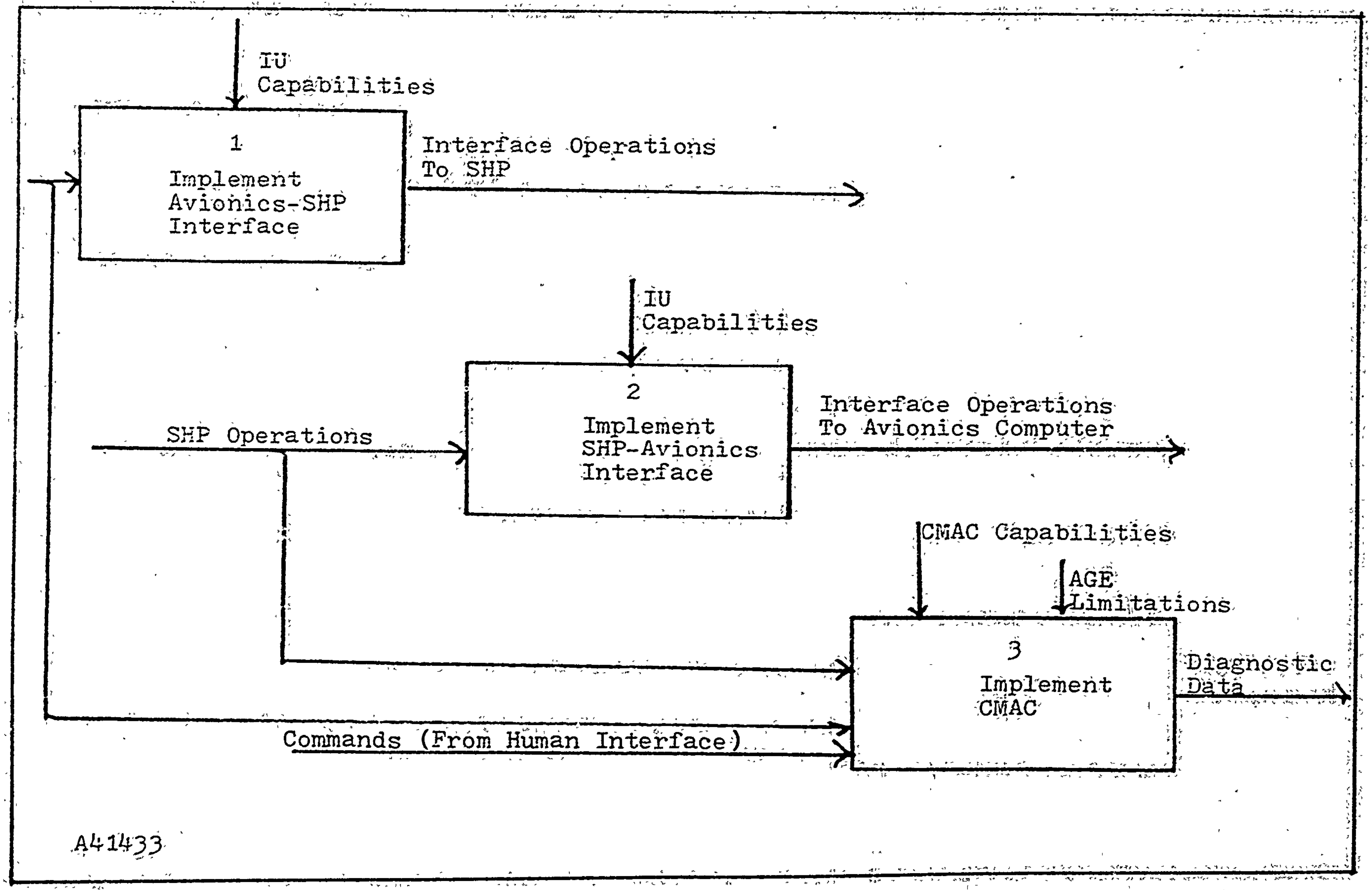

Software Characteristics
Model Execution Commands
Weather/Atmospheric Conditions
Reference Data
Aircraft Position, Velocity, Attitude
2 Simulate Avionic Sensor Outputs
Sensor Outputs
Avionic Sensor Modules
SHP Format Data
Software Characteristics
Weather/Atmosphere
Reference Data
Model Execution Commands
Inputs From IU
1 Scale And Format Module Outputs
Aircraft Position, Velocity, Attitude
Outputs To IU
Model Execution Commands
SHP Format Outputs
3 Provide Inputs To Stimulate Sensors
Sensor Inputs
Support And System Interface Modules
4 Provide Error Generation And Real Time Display
Support Modules

A4143224

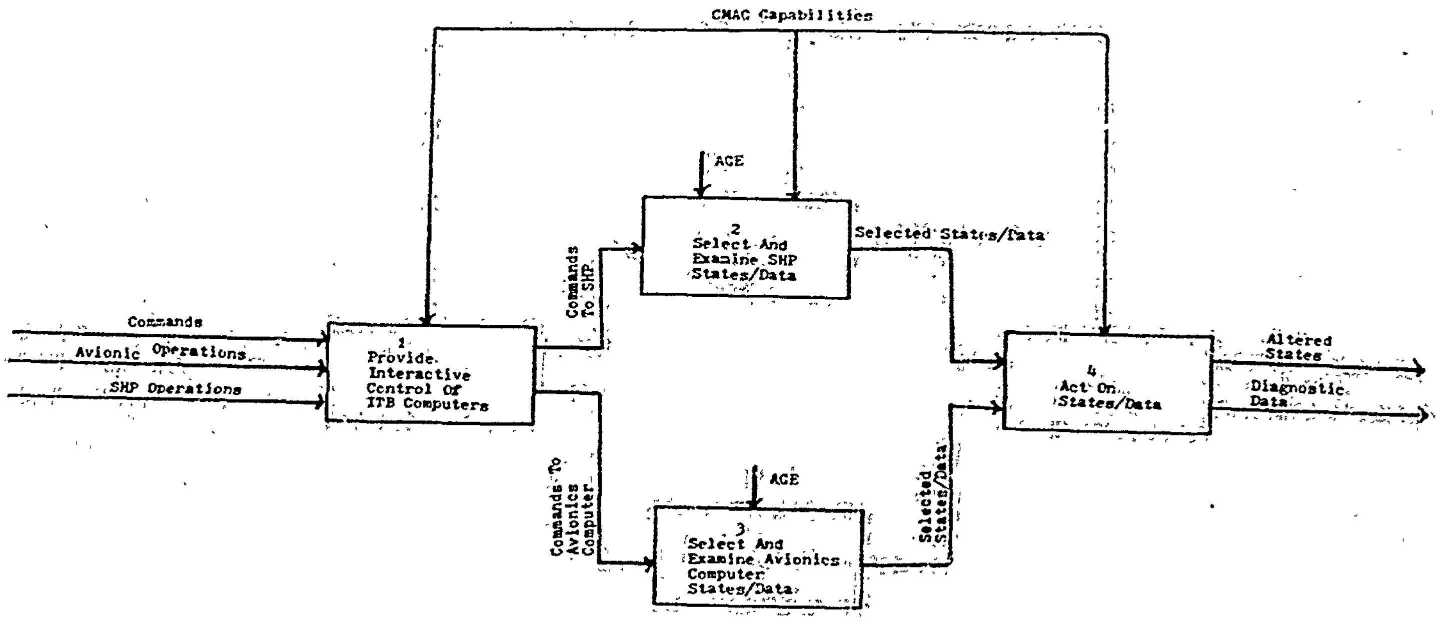
Simulate Sensor Stimuli And Provide Display


Avionics Computer With OFP, Environment, Interface Operations
Time
OFP Capability
Components Available
1 Perform Control/ Display Functions
Control And Display Operations
OFP Capability
Time
Components Available
2 Perform Navigation Functions
Navigation Operations
OFP Capabilities
Time
Components Available
3 Perform Air Data Computations
Air Data Computations
OFP Capabilities
Time
Components Available
4 Radar And Communications
Radar And Communications Operations
OFP Capability
Time
Components Available
6 Munitions Management
Munitions Management Operations
OFP Capability
Time
Components Available
5 Perform Fuels Management
Fuels Management Operations
A414323

Perform Avionics Functions

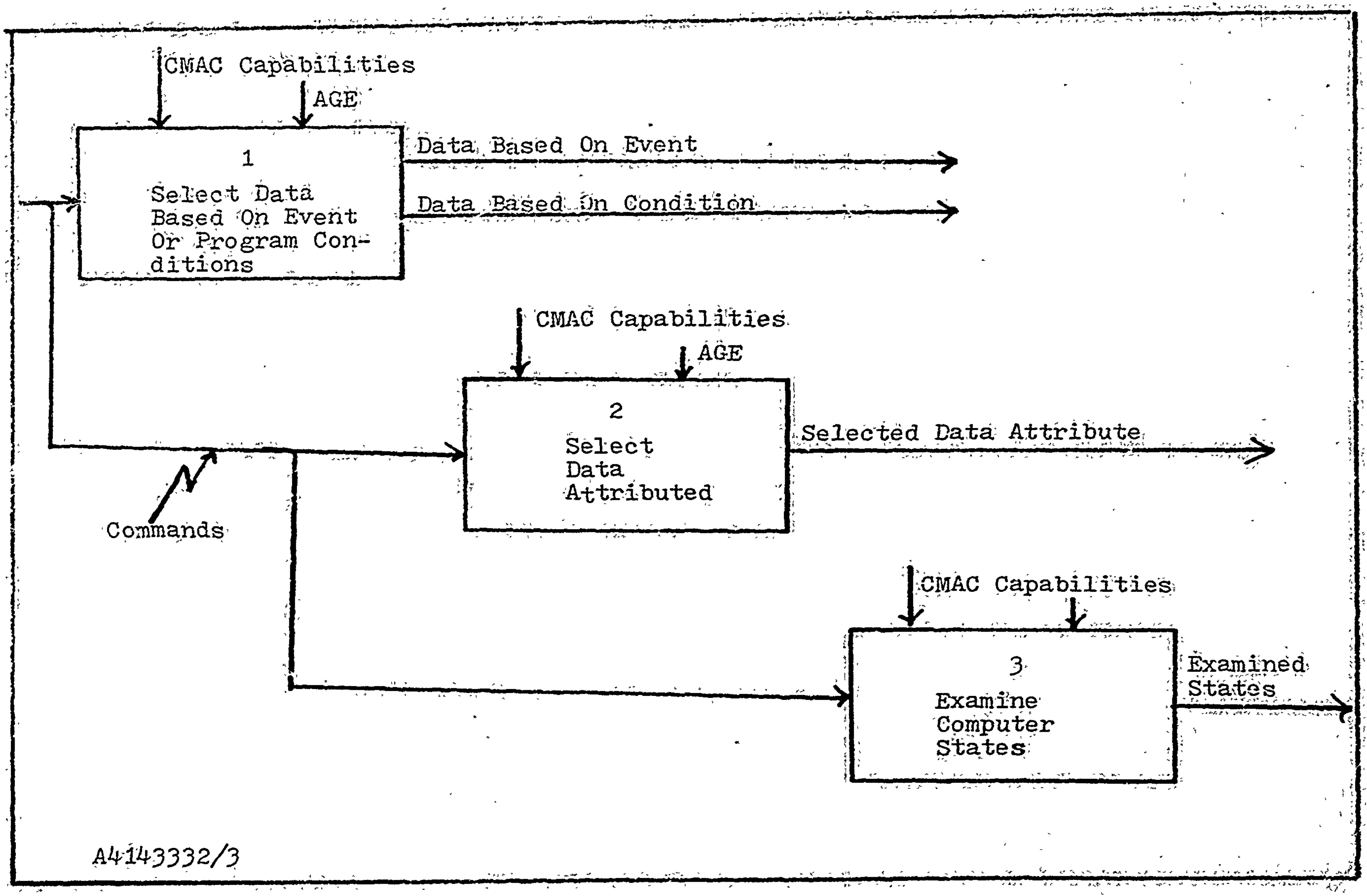


Implement Interface And Diagnostics

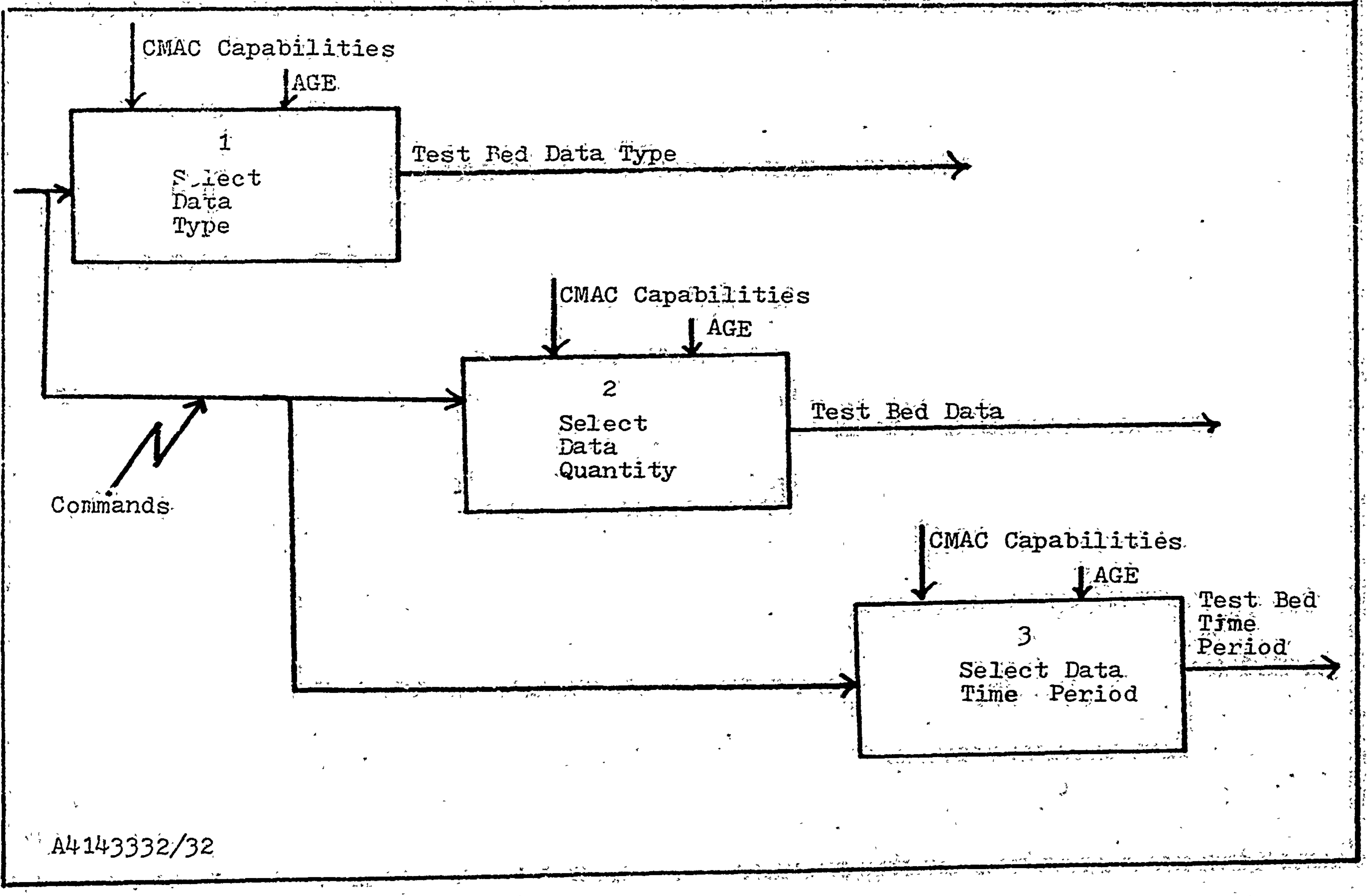


A414333

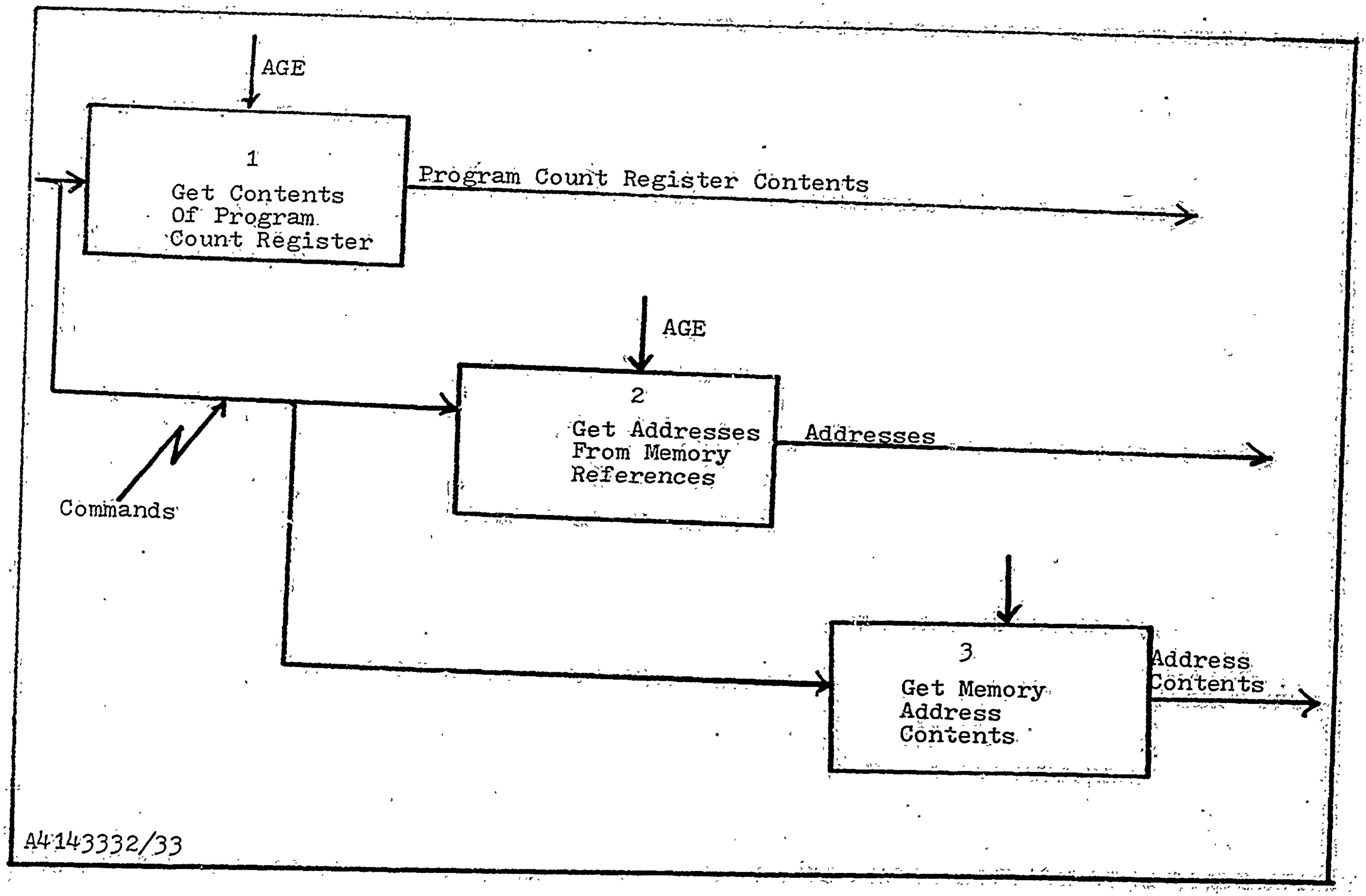
Implement CMAC



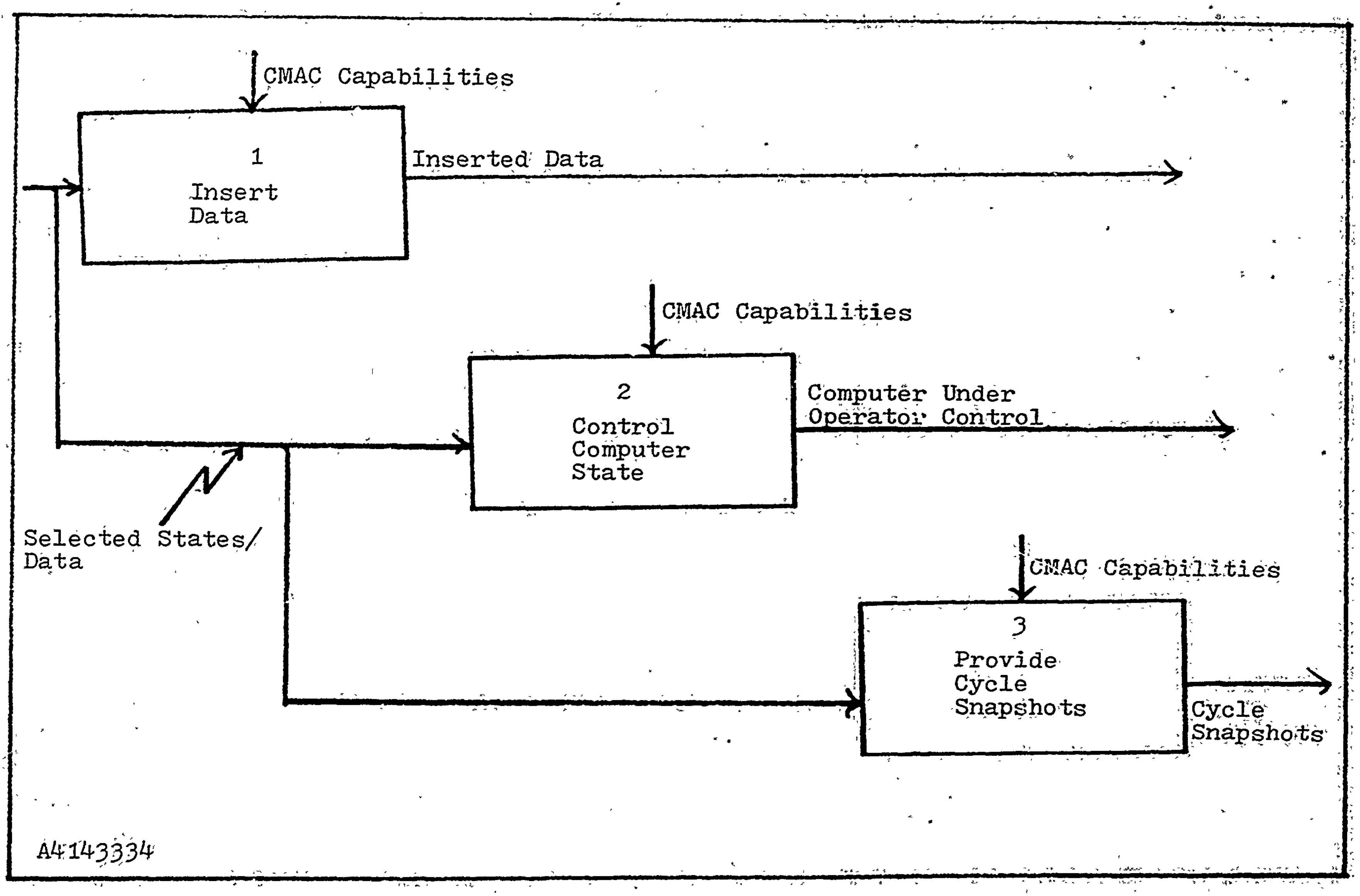
Select And Examine States And Data



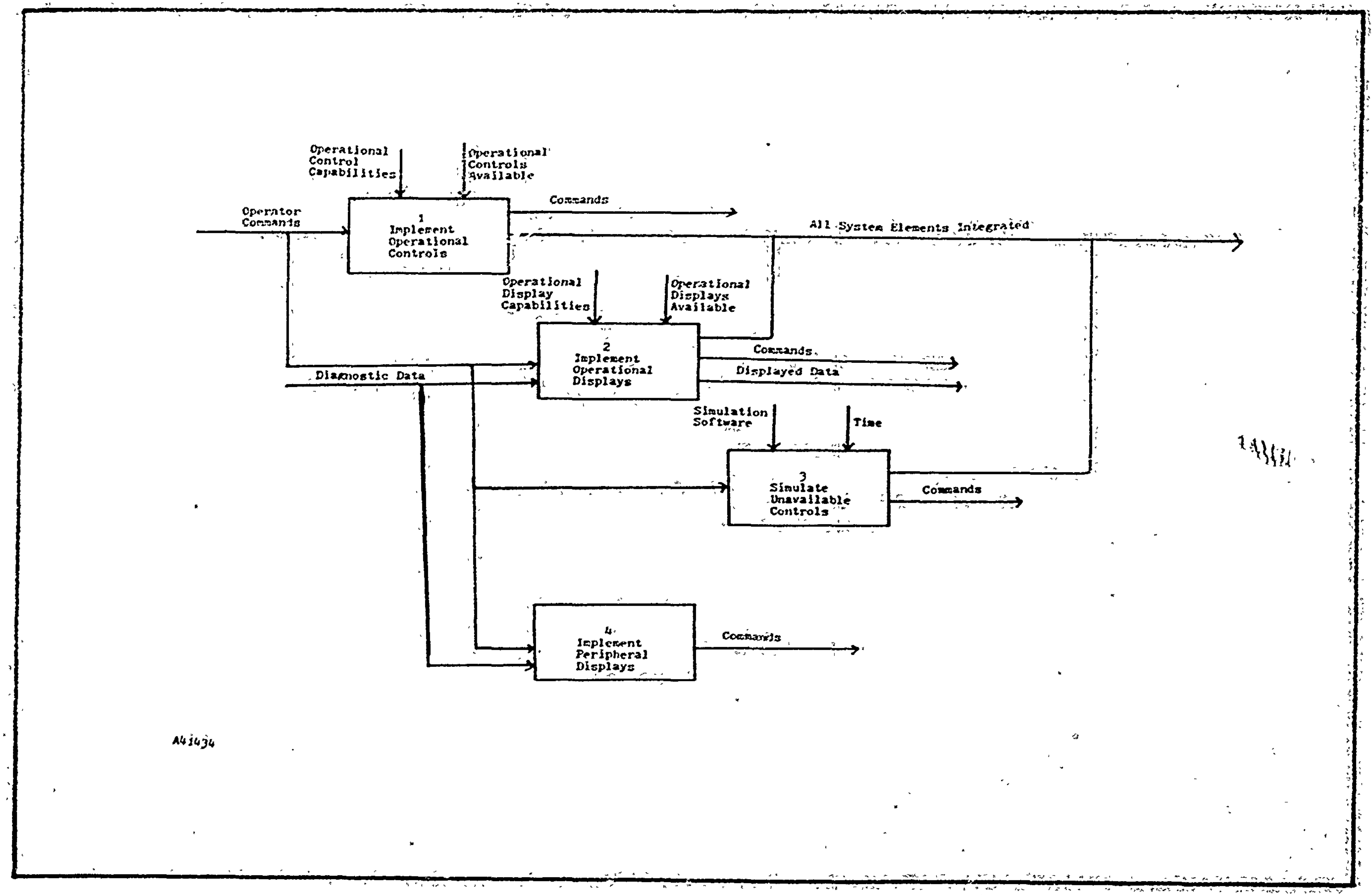
Select Data Attributes



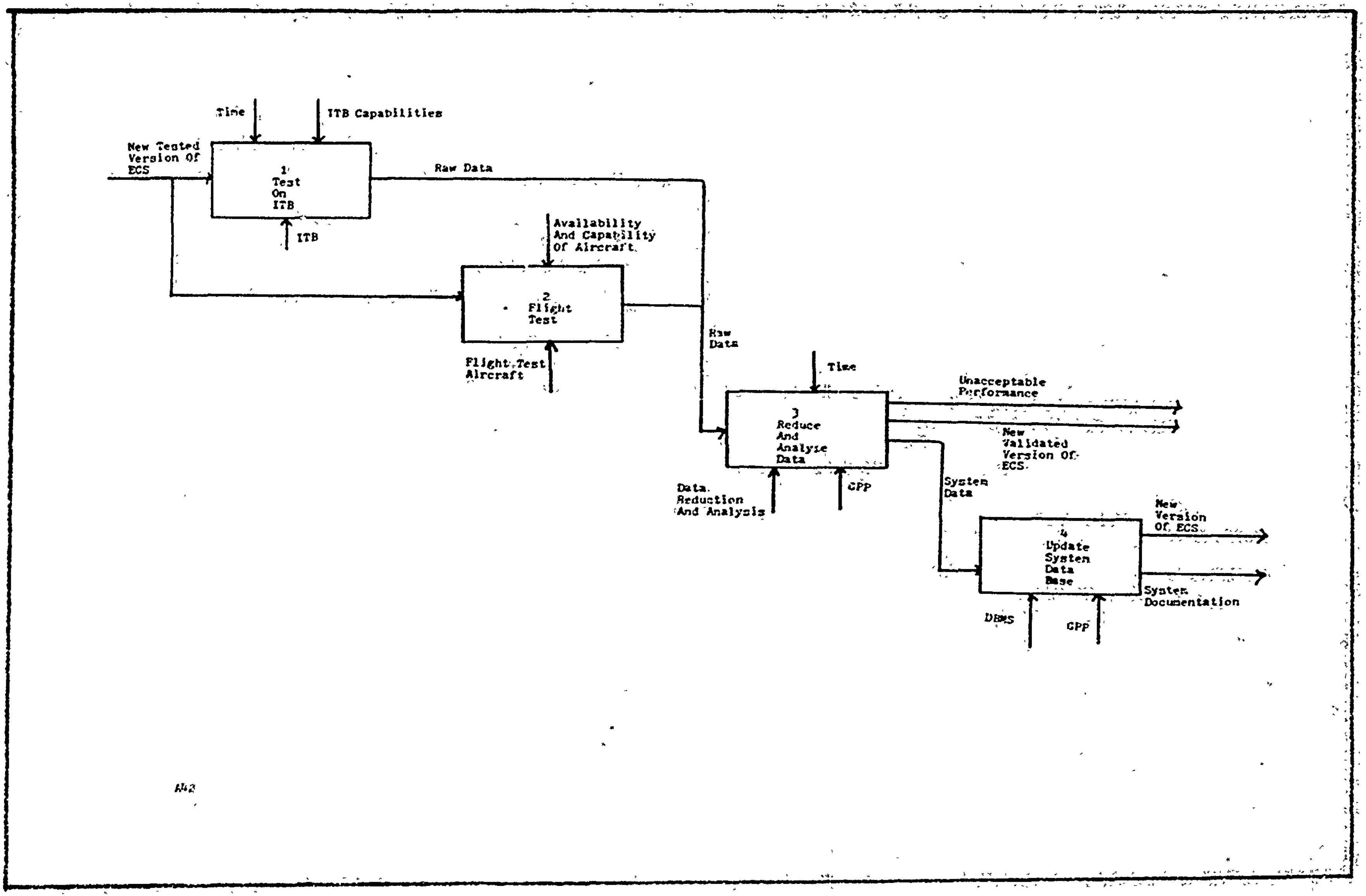
Examine Computer States



Act On States And Data


Operational Control Capabilities
Operational Controls Available
Operator Commands
1 Implement Operational Controls
Commands
All System Elements Integrated
Operational Display Capabilities
Operational Displays Available
2 Implement Operational Displays
Commands
Displayed Data
Diagnostic Data
Simulation Software
Time
3 Simulate Unavailable Controls
Commands
4 Implement Peripheral Displays
Commands
A41434

Implement Human Interface



Accomplish System Acceptance

Appendix D

Q-GERT Tutorial

Queuing Systems Graphical Evaluation and Review Technique (Q-GERT) is a network modeling vehicle and computer analysis tool It was designed for studying the procedural aspects of manufacturing, defense, and service systems. It provides industrial engineers, buisness analysts, and operations researchers with a graphical vehicle for modeling analysis and communication. (Rcf 27:vii)

The design of Q-GERT allows the user to modify or extend his model and allows for hierarchical modeling. Simple "first-cut" models can be constructed quickly, then made more complex. Q-GERT employs an activity-on-branch network philosophy in which a branch represents an activity with a processing time or delay. Nodes separate branches and are used to model milestones, decision points, and queues. The Q-GERT model consists of nodes and branches with transactions flowing through it. The transactions are directed through the network according to the branching characteristics of the nodes. Transactions represent physical objects, information, or, a combination of the two. Different types of nodes allow for modeling complex systems. (Ref 27:3)

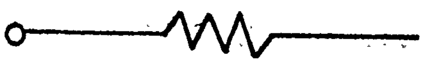
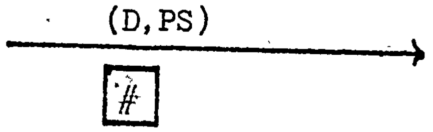
Q-GERT Networks

Transactions originate at source nodes and travel from the start node to the end node of each branch. The transaction continues through the network until no further routing can be performed, where it encounters a sink node or another node which allows for the destruction of information flow. The transactions have attribute values which allow different types of objects to flow through the network and procedures are available to assign and change attribute values of transactions at the various network nodes.

As transactions traverse the network, statistics are collected on travel times via a statistical data collection scheme embedded directly in the Q-GERT network model. The Q-GERT Analysis Program employs a simulation procedure to analyze the network. (Ref 27:4) The network symbols and interconnections provide the following capabilities: the embellishment and compounding of simple systems to form complex ones, a communication mechanism facilitating discussion of significant system features, a means for specifying data requirements to produce a system analysis, and a coputer program for obtaining measures of system performance directly from the network model. (Ref 27:9)

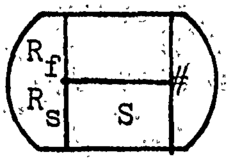
Q-GERT Modeling Symbols.

Provided here is a brief list of Q-GERT symbols used in Chapter VI. A complete, detailed treatment of Q-GERT is found in Reference 27.

Symbol	Use
	Points to a source node
	Activity: D=distribution or function type from which activity time is determined. PS=Parameter set number or constant value where activity time parameters are specified

Symbol Use

= Activity number



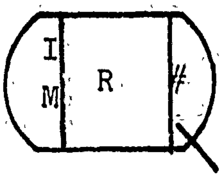
Node:

R_f = Number of incoming transactions to release node for first time.

R_s = Number of incoming transactions to release node all subsequent times.

S = Statistics collection type.

# = Node number



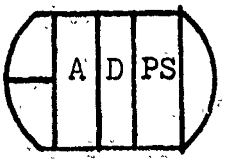
Queue Nodc:

I = Initial number of transactions at Q node.

M = Max. number of transactions in queue.

R = Ranking procedure for ordering transactions.

# = Node number

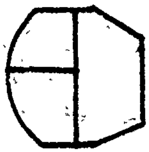


Node With Attribute:

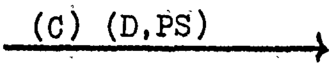
A = Attribute number to which value is to be assigned.

D = Distribution or function type from which assignment value is determined.

PS = Parameter set number.



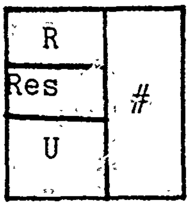
Node with conditional-take-first branching:



Condition specification for a branch:

C = Condition specification for taking the branch

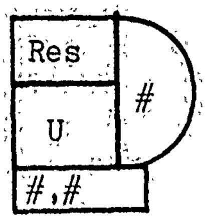
Routing Indicator (No activity):



Allocate Node: Allocation of resources:

R = Allocation rule.

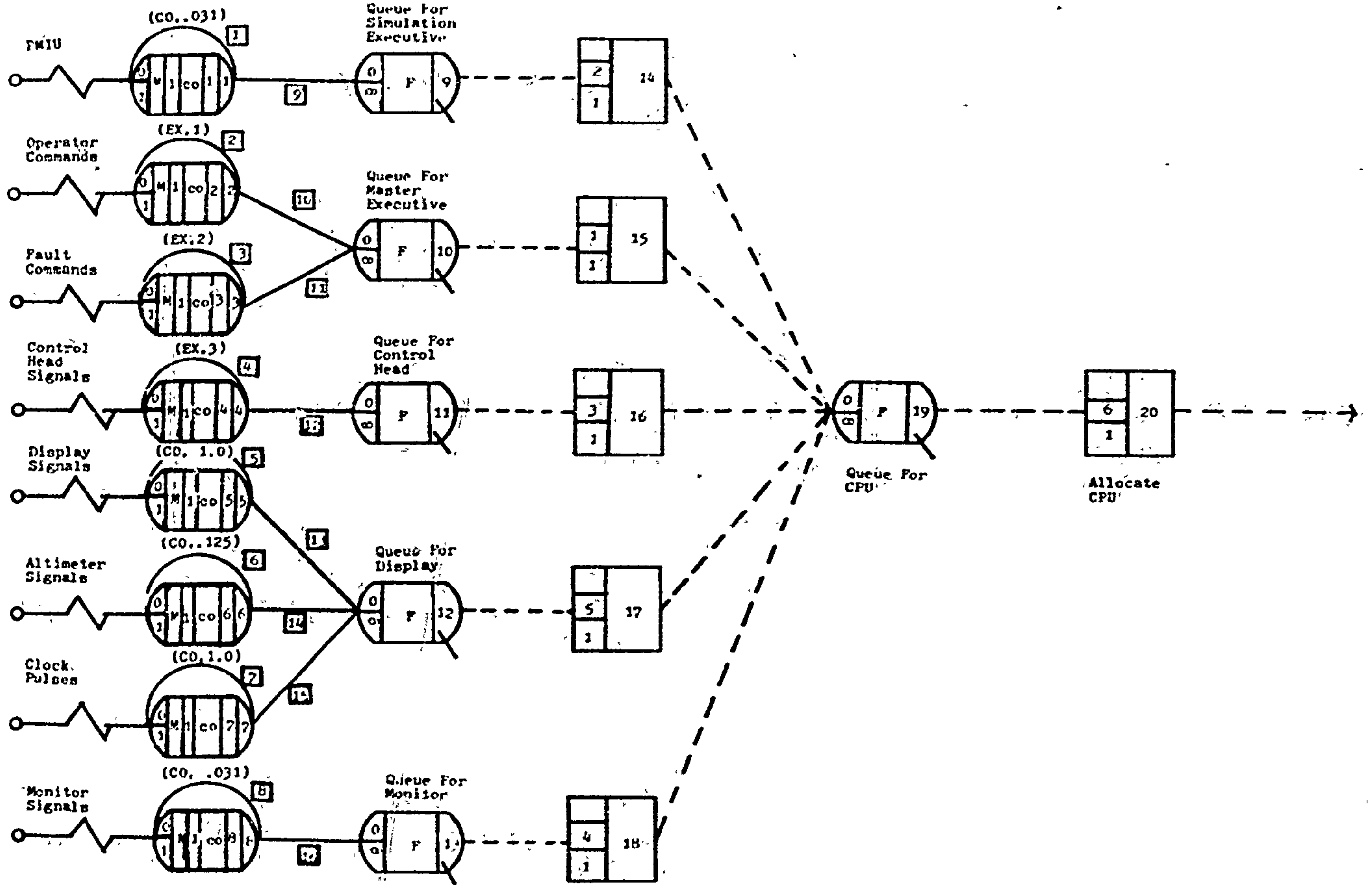
Res = Resource type number to be allocated.

Symbol	Use
	U = Number of resource units to be allocated. # = Node number.
	Free Node: Freeing of previously allocated resources: Res = Resource type number to be freed. U = Number of resource units to free. # = Node number. #,# = Polling order of allocate nodes resource will be returned to.

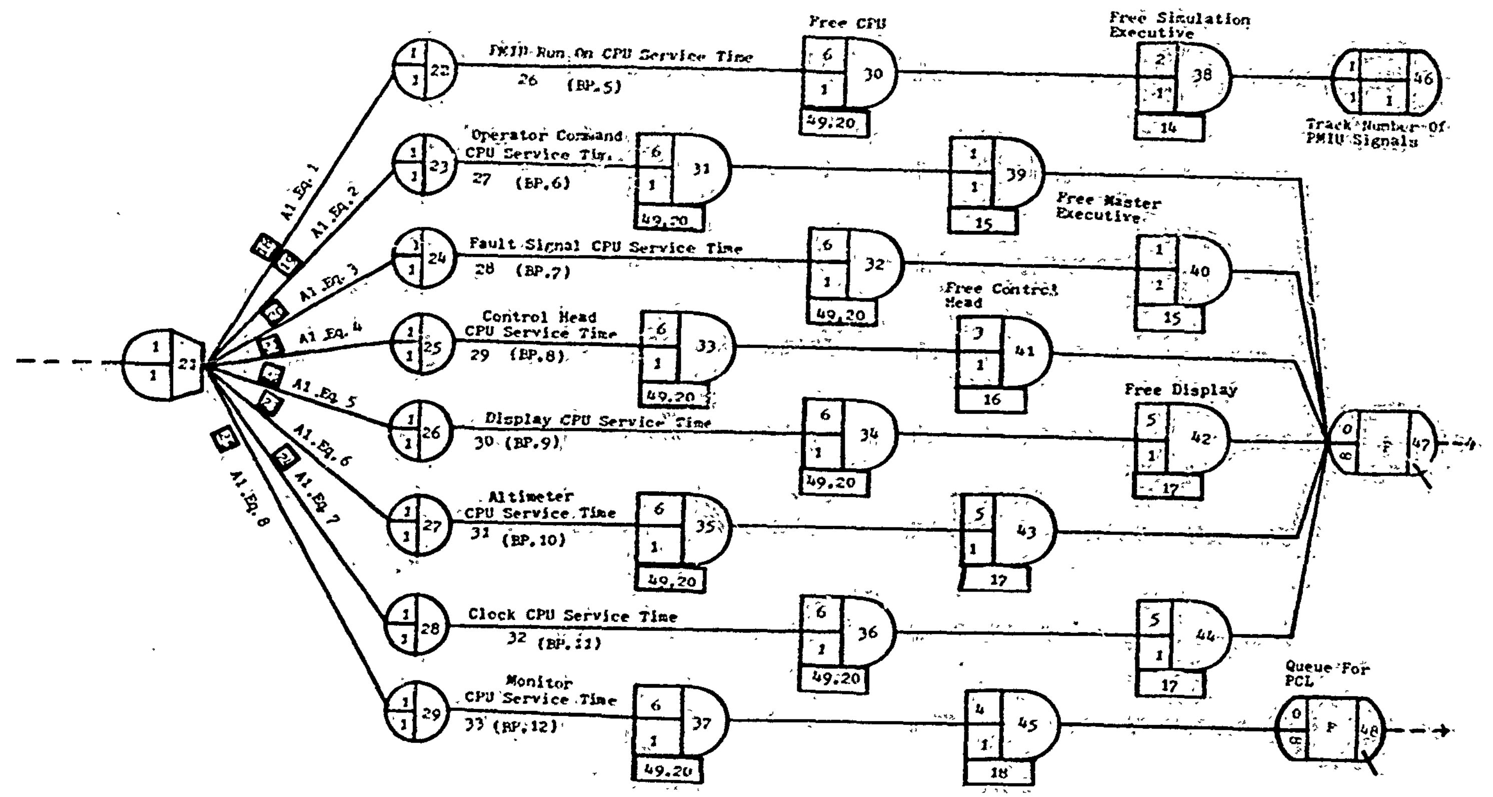
Appendix E

Chapter VI Supporting Data

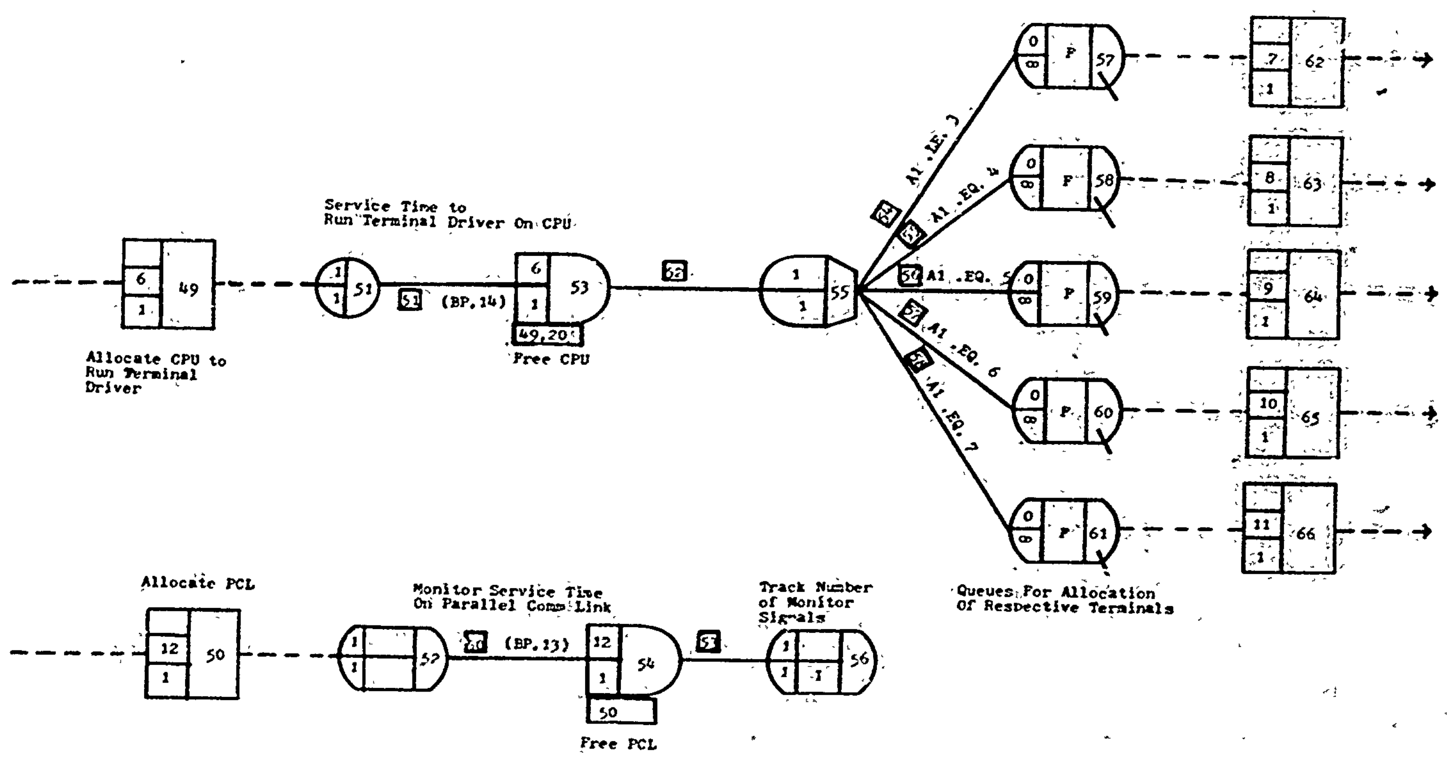
This appendix contains design information and data supporting the experiments and conclusions presented in Chapter VI.



KC-135 AMHB Executive And Control Software Q-GERT Network (Sheet 1 of 4)

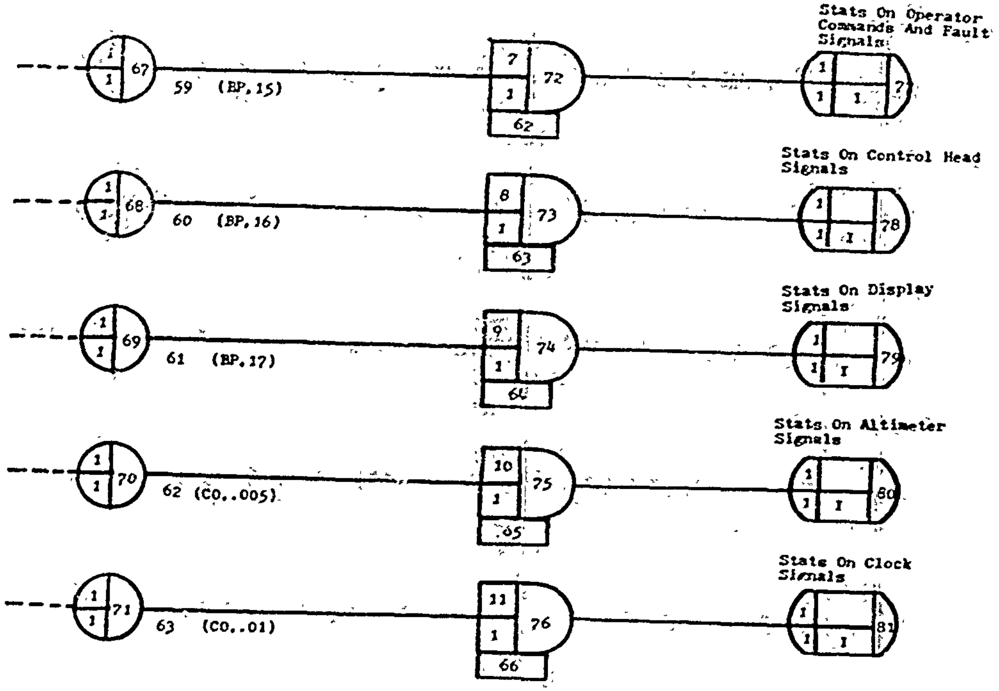


KC-135 AMHB Executive And Control Software Q-GERT Network (Sheet 2 of 4)



Allocate CPU to Run Terminal Driver
49
6
1
Service Time to Run Terminal Driver On CPU
51
(BP,14)
53
49,20
Free CPU
55
A1 .LE. 3
A1 .EQ. 4
A1 .EQ. 5
A1 .EQ. 6
A1 .EQ. 7
0
8
F
57
58
59
60
61
62
63
64
65
66
7
8
9
10
11
Queues For Allocation Of Respective Terminals
Allocate PCL
50
12
Monitor Service Time On Parallel Comm Link
52
(BP,13)
54
Free PCL
Track Number of Monitor Signals
56

KC-135 AMHB Executive And Control Software Q-GERT Network (Sheet 3 of 4)



Resources	
Res #	Resource Value
1	Master Executive Program
2	Simulation Executive Program
3	Control Head Program
4	Monitor Program
5	Display Program
6	VAX CPU
7	Operators Console
8	Control Head Console
9	Display Console
10	Altimeter Console
11	Clock Plasma Display
12	Parallel Communication Link

Attribute #1	
Value	Signal
1	PMIU
2	Operator Command
3	Fault
4	Control Head
5	Display
6	Altimeter
7	Clock
8	Monitor

KC-135 AMHB Executive And Control Software Q-GERT Network (Sheet 4 of 4)

Q-GERT Program Listing

```
GEN,HANSON,PROJECT3,6,1,1981,7,0,0,100,1,,,2*
**********  RESOURCES  **********
RES,1/MASTEREX,1,15*         MASTER EXECUTIVE PROGRAM
RES,2/SIMEXEC,1,14*          SIMULATION EXECUTIVE PROGRAM
RES,3/CONHEAD,1,16*          CONTROL HEAD PROGRAM
RES,4/MONITOR,1,18*          MONITOR PROGRAM
RES,5/DISPLAY,1,17*          DISPLAY PROGRAM
RES,6/CPU,1,49,20*           VAX CPU
RES,7/OPERCNSL,1,62*         OPERATOR'S CONSOLE
RES,8/CHEADCON,1,63*         CONTROL HEAD CONSOLE
RES,9/DISPLCON,1,64*         DISPLAY CONSOLE
RES,10/ALTCON,1,65*          ALTIMETER CONSOLE
RES,11/PLASMA,1,66*          CLOCK PLASMA DISPLAY
RES,12/PCL,1,50*             PARALLEL COMM LINK
**********  NODES  **********
SOU,1,0,1,D,M*           SOURCE FOR PMIU SIGNALS
SOU,2,0,1,D,M*           SOURCE FOR OPERATOR COMMANDS
SOU,3,0,1,D,M*           SOURCE FOR FAULT SIGNALS
SOU,4,0,1,D,M*           SOURCE FOR CONTROL HEAD SIGNALS
SOU,5,0,1,D,M*           SOURCE FOR ALTIMETER SIGNALS
SOU,6,0,1,D,M*           SOURCE FOR DISPLAY SIGNALS
SOU,7,0,1,D,M*           SOURCE FOR CLOCK SIGNALS
SOU,8,0,1,D,M*           SOURCE FOR MONITOR SIGNALS
QUE,9,0,,,F,(10)14*      QUEUE FOR SIMULATION EXEC
QUE,10,0,,,F,(10)15*     QUEUE FOR MASTER EXEC
QUE,11,0,,,F,(10)16*     QUEUE FOR CONTROL HEAD PROGRAM
QUE,12,0,,,F,(10)17*     QUEUE FOR DISPLAY PROGRAM
QUE,13,0,,,F,(10)18*     QUEUE FOR MONITOR PROGRAM
ALL,14,,2,1,9/19*        ALLOCATE SIMULATION EXECUTIVE
ALL,15,,1,1,10/19*       ALLOCATE MASTER EXECUTIVE
ALL,16,,3,1,11/19*       ALLOCATE CONTROL HEAD PROGRAM
ALL,17,,5,1,12/19*       ALLOCATE DISPLAY PROGRAM
ALL,18,,4,1,13/19*       ALLOCATE MONITOR PROGRAM
QUE,19,0,,,F.(10)20*     QUEUE FOR CPU
ALL,20,,6,1,19/21*       ALLOCATE CPU
REG,21,1,1,F*            SELECT SERVICE BASED ON ATT 1
REG,22,1,1,D*
REG,23,1,1,D*
REG,24,1,1,D*
REG,25,1,1,D*
REG,26,1,1,D*
REG,27,1,1,D*
REG,28,1,1,D*
REG,29,1,1,D*
FRE,30,D,6,1,49,20*      FREE CPU
FRE,31,D,6,1,49,20*
FRE,32,D,6,1,49,20*
FRE,33,D,6,1,49,20*
FRE,34,D,6,1,49,20*
FRE,35,D,6,1,49,20*
FRE,36,D,6,1,49,20*
FRE,37,D,6,1,49,20*
FRE,38,D,2,1,14*         FREE SIMULATION EXECUTIVE
```

Q-GERT Program Listing (Continued)

```
FRE,39,D,1,1,15*             FREE MASTER EXECUTIVE
FRE,40,D,1,1,15*             FREE MASTER EXECUTIVE
FRE,41,D,3,1,16*             FREE CONTROL HEAD PROGRAM
FRE,42,D,5,1,17*             FREE DISPLAY PROGRAM
FRE,43,D,5,1,17*             FREE DISPLAY PROGRAM
FRE,44,D,5,1,17*             FREE DISPLAY PROGRAM
FRE,45,D,4,1,18*             FREE MONITOR PROGRAM
STA,46,1,1,,I*               STATISTICS ON PMIU SIGNALS
QUE,47,0,,,F,(10)49*         QUEUE FOR CPU TO RUN TERMINAL DR.
QUE,48,0,,,F,(10)50*         QUEUE FOR PARALLEL COMM LINK
ALL,49,,6,1,47/51*           ALLOCATE CPU TO RUN TERMINAL DR.
ALL,50,,12,1,48/52*          ALLOCATE PARALLEL COMM LINK
REG,51,1,1,D*
REG,52,1,1,D*
FRE,53,D,6,1,49,20*          FREE CPU
FRE,54,D,12,1,50*            FREE PARALLEL COMM LINK
REG,55,1,1,F*                SELECT TERMINAL ACCORDING TO ATT1
STA,56,1,1,,I*               STATISTICS ON MONITOR SIGNALS
QUE,57,0,,,F,(10)62*         QUEUE FOR ALLOCATION OF TERMINALS
QUE,58,0,,,F,(10)63*
QUE,59,0,,,F,(10)64*
QUE,60,0,,,F,(10)65*
QUE,61,0,,,F,(10)66*
ALL,62,,7,1,57/67*           ALLOCATING OPER CMD/FAULT CONSOLE
ALL,63,,8,1,58/68*           ALLOCATING CONTROL HEAD CONSOLE
ALL,64,,9,1,59/69*           ALLOCATING DISPLAY CONSOLE
ALL,65,,10,1,60/70*          ALLOCATING ALTIMETER CONSOLE
ALL,66,,11,1,61/71           ALLOCATING PLASMA DISPLAY
REG,67,1,1,D*
REG,68,1,1,D*
REG,69,1,1,D*
REG,70,1,1,D*
REG,71,1,1,D*
FRE,72,D,7,1,62*             FREE OPER CMD/FAULT CONSOLE
FRE,73,D,8,1,63*             FREE CONTROL HEAD CONSOLE
FRE,74,D,9,1,64*             FREE DISPLAY CONSOLE
FRE,75,D,10,1,65*            FREE ALTIMETER CONSOLE
FRE,76,D,11,1,65*            FREE PLASMA DISPLAY
STA,77,!,!,,I*               STATISTICS ON OPER CMD/FLT SIGS
STA,78,1,1,,I*               STATISTICS ON CONTROL HEAD SIGS
STA,79,1,1,,I*               STATISTICS ON DISPLAY SIGS
STA,80,1,1,,I*               STATISTICS ON ALTIMETER SIGS
STA,81,1,1,,I*               STATISTICS ON CLOCK SIGS
**********  VAS CARDS  **********
VAS,1,1,CO,1*
VAS,2,1,CO,2*
VAS,3,1,CO,3*
VAS,4,1,CO,4*
VAS,5,1,CO,5*
VAS,6,1,CO,6*
VAS,7,1,CO,7*
VAS,8,1,CO,8*
**********  PARAMETER CARDS  **********
PAR,1,420.0,240.0,600.0*     INTERARRIVAL TIME-OPER COMMANDS
```

Q-GERT Program Listing (Continued)

```
PAR,2,300.0,60.0,540.0*    FAULT SIGNAL INTERARRIVAL TIME
PAR,3,120.0,60.0,300.0*    CONTROL HEAD SIGNAL INER. TIME
PAR,4,0.125,0.0625,1.0*    ALTIMETER INTERARRIVAL TIME
PAR,5,0.006,0.001,0.012*   PMIU CPU SERVICE TIME
PAR,6,0.0005,0.0001,0.0010* OPER COMMAND CPU SERVICE TIME
PAR,7,0.0005,0.0001,0.0010* FAULT CPU SERVICE TIME
PAR,8,0.0005,0.0001,0.0010* CONTROL HEAD CPU SERVICE TIME
PAR,9,0.0005,0.0001,0.0010* DISPLAY CPU SERVICE TIME
PAR,10,0.00025,0.0001,0.0005*   ALTIMETER CPU SERVICE TIME
PAR,11,0.00025,0.0001,0.0005*   PLASMA CLOCK CPU SERVICE TIME
PAR,12,0.001,0.00025,0.002*     MONITOR CPU SERVICE TIME
PAR,13,0.001,0.00025,0.002*     PCL TRANSMISSION SVC TIME
PAR,14,0.00010,.00005,0.00025*  TERMINAL DRIVER SVC TIME
PAR,15,0.500,0.100,1.0*    OPER CMD/FAULT TO CONSOLE SVC TIME
PAR,16,0.500,0.100,1.0*    SVC TIME TO OUTPUT CONHD TO CONS
PAR,17,0.500,0.100,1.0*    SVC TIME TO OUTPUT DISPLAY
PAR,18,0.200,0.100,0.400*  SVC TIME TO OUTPUT ALTIMETER
PAR,19,0.200,0.100,0.400*  SVC TIME TO OUTPUT CLOCK TO PLSMA
**********  ACTIVITY  CARDS  **********
ACT,1,1,CO,0.031,1*        INTERARRIVAL OF PMIU SIGNALS
ACT,2,2,EX,1,2*            INTERARRIVAL OF OPERATOR CMDS
ACT,3,3,EX,2,3*            INTERARRIVAL OF FAULT SIGNALS
ACT,4,4,EX,3,4*            INTERARRIVAL OF CONTROL HEAD SIGS
ACT,5,5,CO,1.0,5*          INTERARRIVAL OF DISPLAY SIGNALS
ACT,6,6,EX,4,6*            INTERARRIVAL OF ALTIMETER SIGS
ACT,7,7,CO,1.0,7*          INTERARRIVAL OF CLOCK SIGNALS
ACT,8,8,CO,0.031,8*        INTERARRIVAL OF MONITOR SIGS
ACT,1,9,(6)9*
ACT,2,10,(6)10*
ACT,3,10,(6)11*
ACT,4,11,(6)12*
ACT,5,12,(6)13*
ACT,6,12,(6)14*
ACT,7,12,(6)15*
ACT,8,13,(6)16*
ACT,21,22,,,18,,,A1.EQ.1*
ACT,21,23,,,19,,,A1.EQ.2*
ACT,21,24,,,20,,,A1.EQ.3*
ACT,21,25,,,21,,,A1.EQ.4*
ACT,21,26,,,22,,,A1.EQ.5*
ACT,21,27,,,23,,,A1.EQ.6*
ACT,21,28,,,24,,,A1.EQ.7*
ACT,21,29,,,25,,,A1.EQ.8*
ACT,22,30,BP,5,26*    SERVICE TIME TO RUN PMIU ON CPU
ACT,23,30,BP,6,27*    SERVICE TIME TO RUN OPER CMD ON CPU
ACT,24,32,BP,7,28*    SERVICE TIME TO RUN FAULT ON CPU
ACT,25,33,BP,8,29*    SERVICE TIME TO RUN CNHEAD ON CPU
ACT,26,34,BP,9,30*    SERVICE TIME TO RUN DSPLAY ON CPU
ACT,27,35,BP,10,31*   SERVICE TIME TO RUN ALT ON CPU
ACT,28,36,BP,11,32*   SERVICE TIME TO RUN CLOCK ON CPU
ACT,29,37,BP,12,33*   SERVICE TIME TO RUN MONITOR ON CPU
ACT,30,38,(6)34*
ACT,31,39,(6)35*
ACT,32,40,(6)36*
```

Q-GERT Program Listing (Continued)

```
ACT,33,41,(6)37*
ACT,34,42,(6)38*
ACT,35,43,(6)39*
ACT,36,44,(6)40*
ACT,37,45,(6)41*
ACT,38,46,(6)42*
ACT,39,47,(6)43*
ACT,40,47,(6)44*
ACT,41,47,(6)45*
ACT,42,47,(6)46*
ACT,43,47,(6)47*
ACT,44,47,(6)48*
ACT,45,48,(6)49*
ACT,52,54,BP,13,50*   MONITOR SERVICE TIME ON PCL
ACT,51,53,BP,14,51*   TERMINAL DRIVER SERVICE TIME ON CPU
ACT,53,55,(6)52*
ACT,54,56,(6)53*
ACT,55,57,,,54,,,A1.LE.3*
ACT,55,58,,,55,,,A1.EQ.4*
ACT,55,59,,,56,,,A1.EQ.5*
ACT,55,60,,,57,,,A1.EQ.6*
ACT,55,61,,,58,,,A1.EQ.7*
ACT,67,72,BP,15,59*   OUTPUT OF OPER CMD OR FAULT TO CONSOLE
ACT,68,73,BP,16,60*   OUTPUT OF CONTROL HEAD TO CONSOLE
ACT,69,74,BP,17,61*   OUTPUT OF DISPLAY SIGNAL TO CONSOLE
ACT,70,75,BP,18,62*   OUTPUT OF ALTIMETER TO CONSOLE
ACT,71,76,BP,19,63*   OUTPUT OF CLOCK VALUE TO PLASMA DISPLAY
ACT,72,77,(6)64*
ACT,73,78,(6)65*
ACT,74,79,(6)66*
ACT,75,80,(6)67*
ACT,76,81,(6)68*
FIN*
```

Design Configurations

Experiments 1&2:

	Run Time	Interarrival Times
1.	10 Sec	Original
2.	100 Sec	Original
3.	100 Sec	Updated

Experiment 3:

	Run Time/ Seconds	Parameter Modified	Times Increased
1.	100	14	X3
2.	100	14	X6
3.	100	14	X15
4.	100	14	X50
5.	100	14	X75
6.	100	14	X150
7.	100	14	X300
8.	100	14	X350
9.	100	14	X375
10.	100	14	X400
11.	100	5	X2
12.	100	5	X3
13.	100	5	X4
14.	100	5	X4.5
15.	100	5	X5
16	100	5	X10
17.	100	5	X100
18.	100	6	X10
19.	100	6	X50
20.	100	6	X100
21.	100	6	X200
22.	100	6	X300
23.	100	6	X350
24.	100	6	X400
25.	100	10	X100

Experiment 3 (Continued):

	Run Time/ Seconds	Parameter Modified	Times Increased
26.	100	10	X200
27.	100	10	X300
28.	100	10	X310
29.	100	12	X2
30.	100	12	X10
31.	100	12	X17
32.	100	12	X25

Experiment 4:

	Run Time/ Seconds	Priority Scheme	Parametric Model
1.	100	1	Original
2.	100	1	Max. Term. Driver
3.	100	1	Max. PMIU
4.	100	1	Max. O.C., Cnt. Hd., Fault, Display
5.	100	1	Max. Altimeter
6.	100	1	Max. Monitor
7.	100	2	Original
8.	100	2	Max. Ter. Driver
9.	100	2	Max. PMIU
10.	100	2	Max O.C., Cnt. Hd., Fault, Display
11.	100	2	Max. Altimeter
12.	100	2	Max. Monitor
13.	100	3	Original
14.	100	3	Max. PMIU
15.	100	3	Max. O.C., Cnt.Hd., Fault, Display
16.	100	3	Max. Monitor
17.	100	3	Max. Altimeter
18.	100	3	Max. Ter. Driver

Experiment 5:

	Run Time/ Seconds	System Design	Parametric Model
1.	100	Altered	Original

Experiment 5 (Continued):

	Run Time/ Seconds	System Design	Parametric Model
2.	100	Altered	Max. Term. Driver
3.	100	Altered	Max. PMIU
4.	100	Altered	Max. O.C., Con. Hd., Fault, Display
5.	100	Altered	Max. Altimeter
6.	100	Altered	Max. Monitor

Statistical Results

Experiment 3

CPU Run Time					TIS					
Signal	.0003	.0006	.0015	0.005	.0075	0.015	0.030	0.035	0.037	0.040
Clock	.0127	.0141	.0155	.0239	.0301	.0483	.0833	.0973	.0988	.1076
Altimeter	.0071	.0080	.0084	.0142	.0180	.0296	.0523	.0618	.0655	.0701
Display	.5163	.5213	.5052	.5193	.5381	.5115	.5731	.5608	.5773	.5589
Con. Hd.	.5370	.5388	.5407	.5513	.5589	.5187	.6273	.6425	.6501	.6577
OC/Fault	.5583	.5590	.5597	.5639	.5669	.6275	.6713	.5703	.6968	.6998
Monitor	.0082	.0083	.0083	.0086	.0089	.0112	.0222	.0304	.0331	.0391
PMIU	.0062	.0062	.0062	.0064	.0067	.0086	.0188	.0265	.0292	.0349

"Increased CPU Run Time for Terminal Driver"

CPU Run Time					
Signal	0.012	0.018	0.024	0.027	0.030
Clock	.0145	.0181	.0428	.0596	.0918
Altimeter	.0093	.0135	.0285	.0447	.0615
Display	.5249	.5264	.5293	.5344	.5966
Con. Hd.	.5439	.5514	.5588	.5626	.5663
OC/Fault	.5056	.6175	.6250	.6287	.5468
Monitor	.0144	.0206	.0289	.0428	1.9992
PMIU	.0123	.0185	.0268	.0407	1.9770

" Increased CPU Run Time for PMIU"

CPU Run Time							
Signal	0.005	0.025	0.050	0.100	0.150	0.175	0.200
Clock	.0179	.0447	.0772	.1305	.1887	.2137	.2386
Altimeter	.0075	.0109	.0153	.0252	.0403	.0480	.0567
Display	.5321	.5534	.5826	.6360	.7037	.7383	.7458
Con. Hd.	.5475	.5970	.6588	.7824	.9060	.9679	1.0297
OC/Fault	.5692	.6417	.5982	.8468	1.0536	1.142	1.2305
Monitor	.0083	.0087	.0103	.0166	.0284	.0359	.0443
PMIU	.0062	.0066	.0081	.0143	.0261	.0336	.4190

" Increased CPU Run Time for Oper. Cmd., Fault, Display, and Con. Head"

Experiment 3

TIS

CPU Run Time Signal	0.025	0.050	0.075	.0775
Clock	.0395	.0952	.0626	.1056
Altimeter	.0429	.1144	.0745	.1250
Display	.5464	.5398	.5657	.5455
Con. Hd.	.5364	.5364	.5364	.5364
OC/Fault	.5581	.5278	.6688	.5278
Monitor	.0122	.0168	.0288	.3420
PMIU	.0101	.0165	.0266	.3376

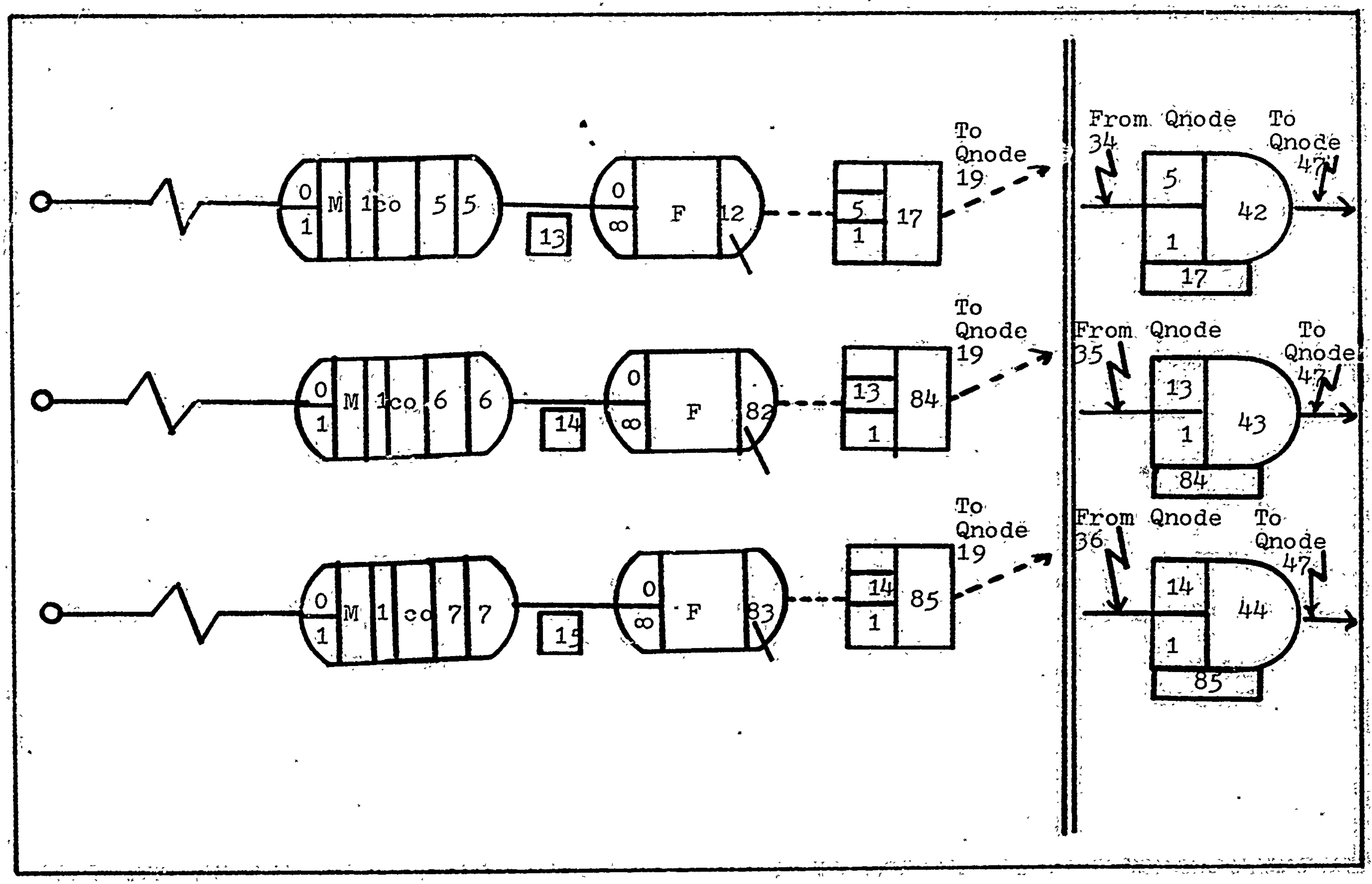
"Increased CPU Run Time for Altimeter and Clock"

CPU Run Time Signal	0.002	0.0100	0.017	0.025
Clock	.0125	.0160	.0259	.0887
Altimeter	.0071	.0107	.0171	.0584
Display	.5164	.5238	.5205	.5436
Con. Hd.	.5364	.5364	.5364	.5364
OC/Fault	.5581	.5795	.5795	.5501
Monitor	.0093	.0176	.0254	2.045
PMIU	.0062	.0062	.0064	2.019

"Increased CPU Run Time for Monitor"

Experiment 4

Parametric Signal	Model Original	Max. Term. Driver	Max. PMIU	Max.OC/Fault/ Con Hd/Display	Max. Alt/Clock	Max. Monitor	
Clock	.0122	.1017	.0539	.2230	.1169	.0238	
Altimeter	.0068	.0716	.0454	.0483	.1594	.0200	
Display	.5133	.5994	.5939	.6820	.6005	.5305	
Con. Hd.	.5364	.6501	.5588	.9060	.5364	.5364	Priority Scheme 1
Oc/Fault	.5708	.4132	.6499	.8361	.6211	.5708	
Monitor	.0082	.0240	.0294	.0267	.0470	.0252	
PMIU	.0061	.0219	.0273	.0245	.0449	.0064	
Clock	.0122	.1011	.0443	.2299	.1088	.0238	
Altimeter	.0068	.0714	.0434	.0519	.1480	.0200	
Display	.5153	.5980	.5766	.7056	.5499	.5325	
Con. Hd.	.6086	.6005	.5027	1.1988	.6086	.6252	Priority Scheme 2
OC/Fault	.3921	.6376	.5598	1.0676	.5467	.5293	
Monitor	.0082	.0240	.0292	.0294	.1579	.0252	
PMIU	.0061	.0218	.0270	.0252	.0440	.0064	
Clock	.0122	.1017	.0539	.2230	.1169	.0238	
Altimeter	.0068	.0716	.0454	.0483	.1514	.0200	
Display	.5133	.5994	.5939	.6820	.6005	.5303	Priority Scheme 3
Con. Hd.	.5364	.6501	.5588	.9060	.5364	.5364	
OC/Fault	.5708	.4132	.6499	.8361	.6211	.5708	
Monitor	.0082	.0290	.0294	.0267	.0440	.0252	
PMIU	.0061	.0219	.0273	.0245	.0449	.0064	

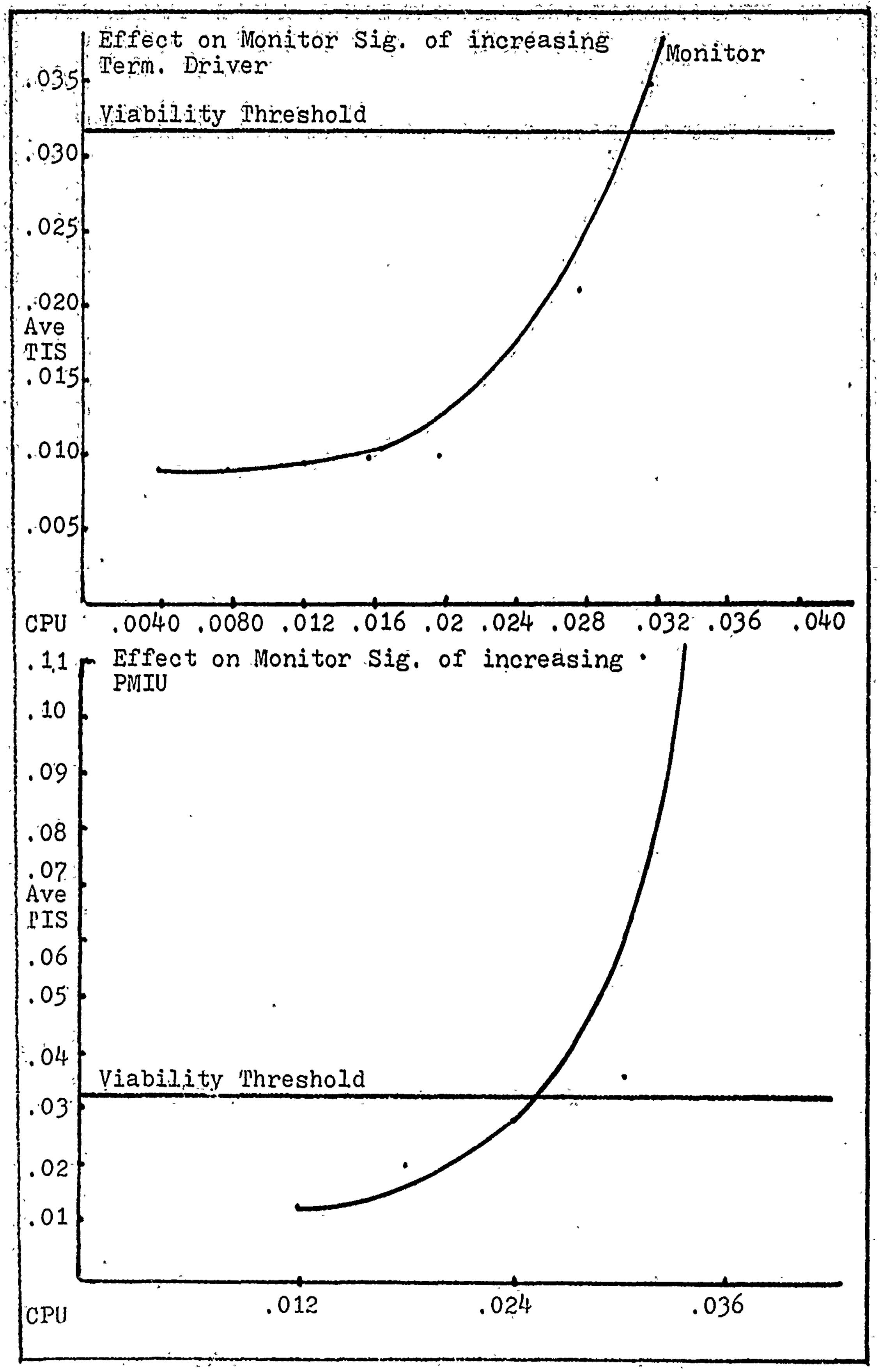


Experiment 5:Q-GERT Network Changes

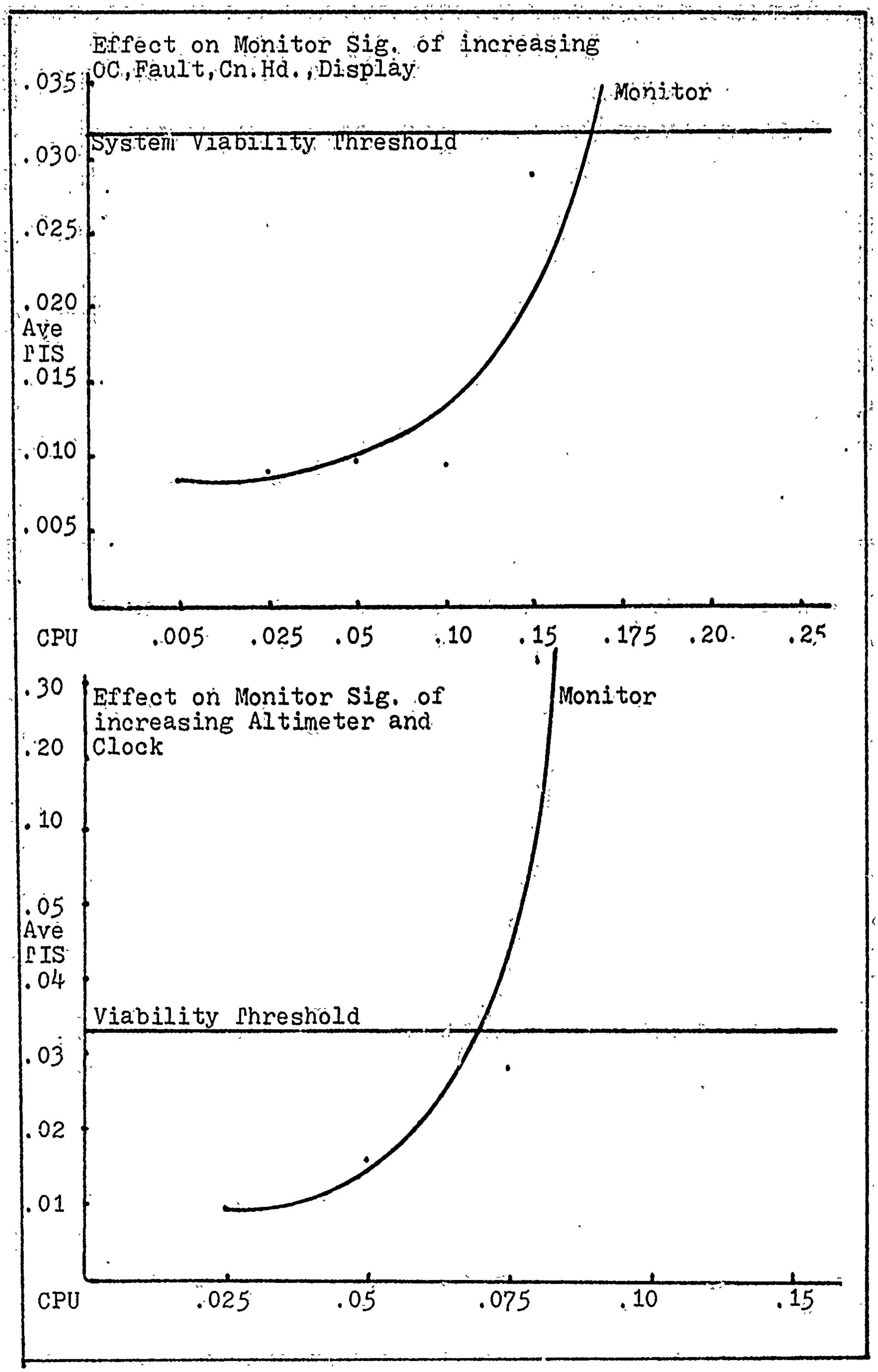
Experiment 5

Parametric Signal	Model Original	Max. Term. Driver	Max. PMIU	Max. Con.	OC/Fault/ Hd./Display	Max. Alt/Clock	Max. Monitor
Clock	.0122	.0984	.0242		.1737	.0933	.0219
Altimeter	.0068	.0650	.0276		.0364	.1110	.0166
Dis;oay	.5161	.5705	.5490		.6861	.6167	.5287
Con. Hd.	.5364	.6501	.5588		.9060	.5364	.5364
OC/Fault	.5708	.5918	.6250		.9273	.5789	.5795
Monitor	.0082	.0328	.0294		.0283	.1711	.0252
PMIU	.0061	.0288	.0273		.0260	.1673	.0064

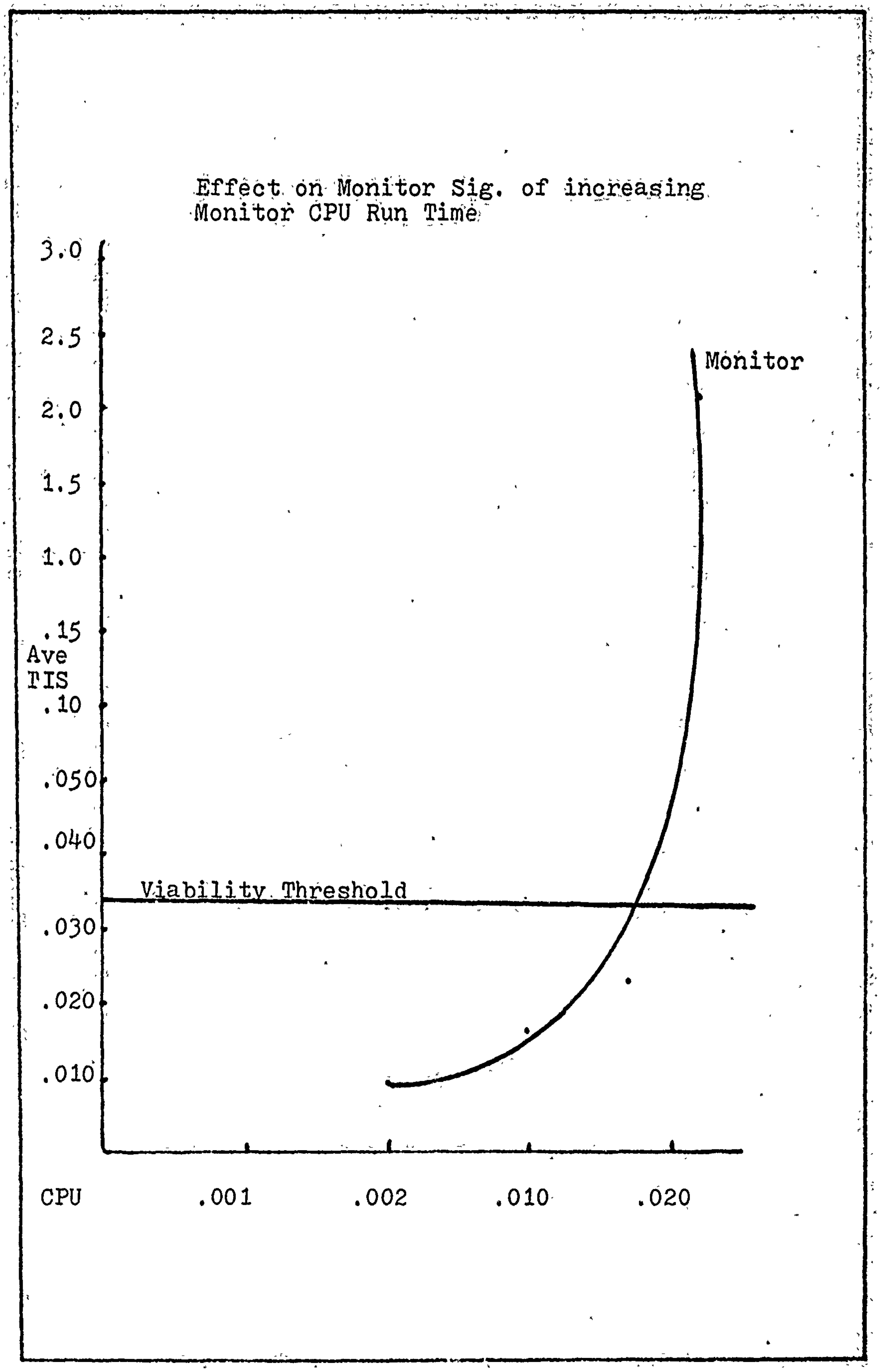
Results Of Experiment Three



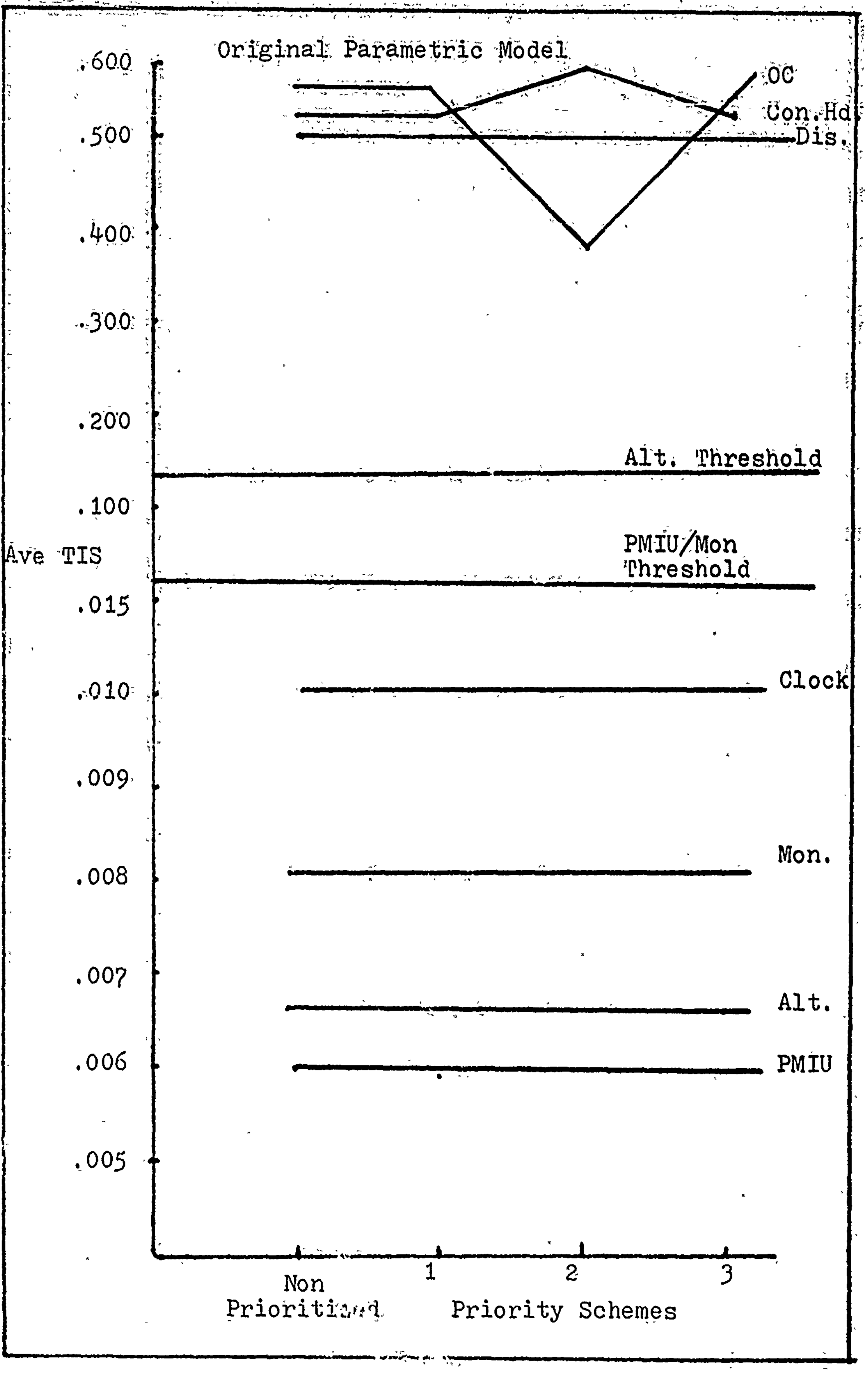
Results Of Experiment Three



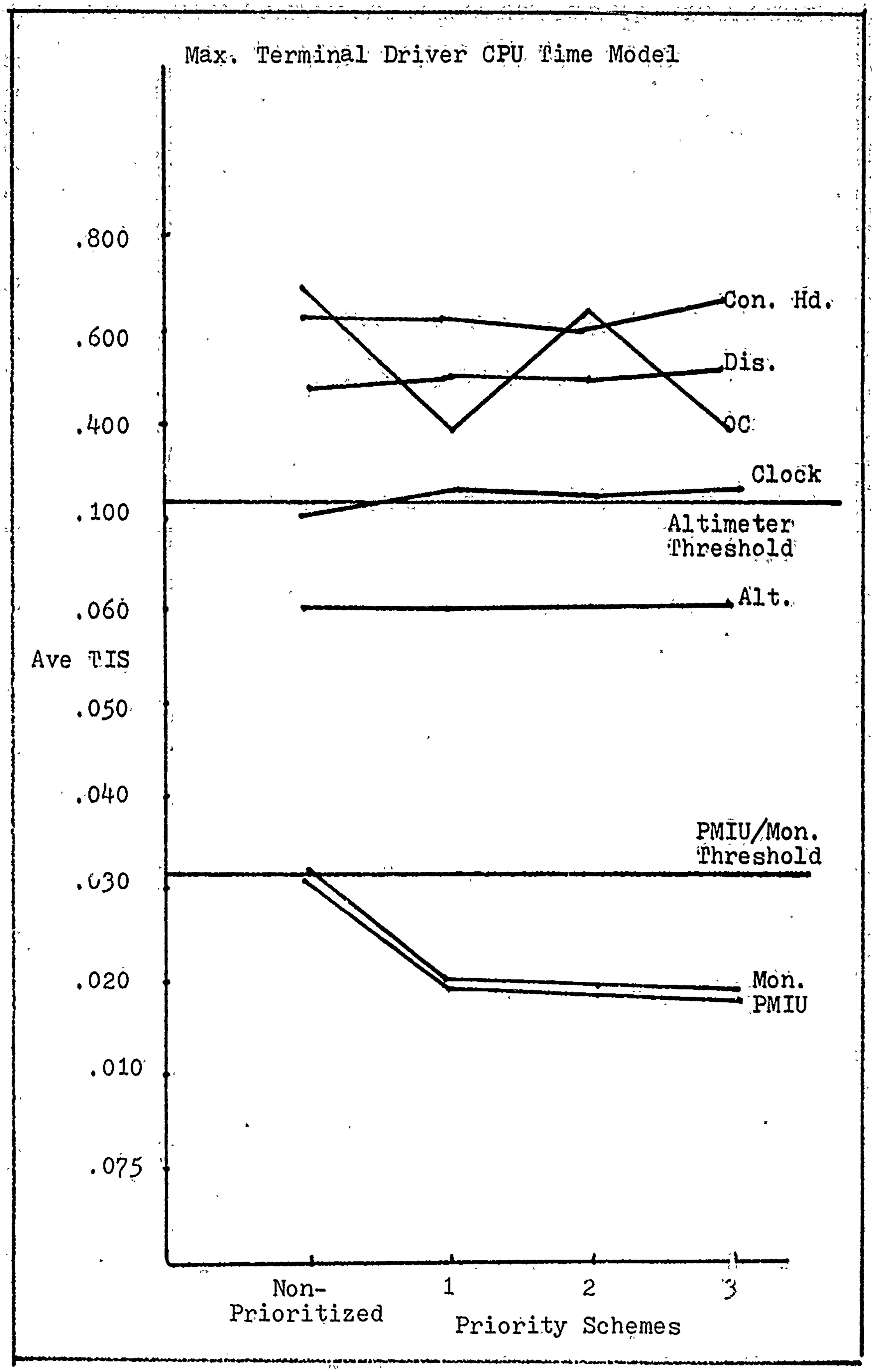
Results Of Experiment Three



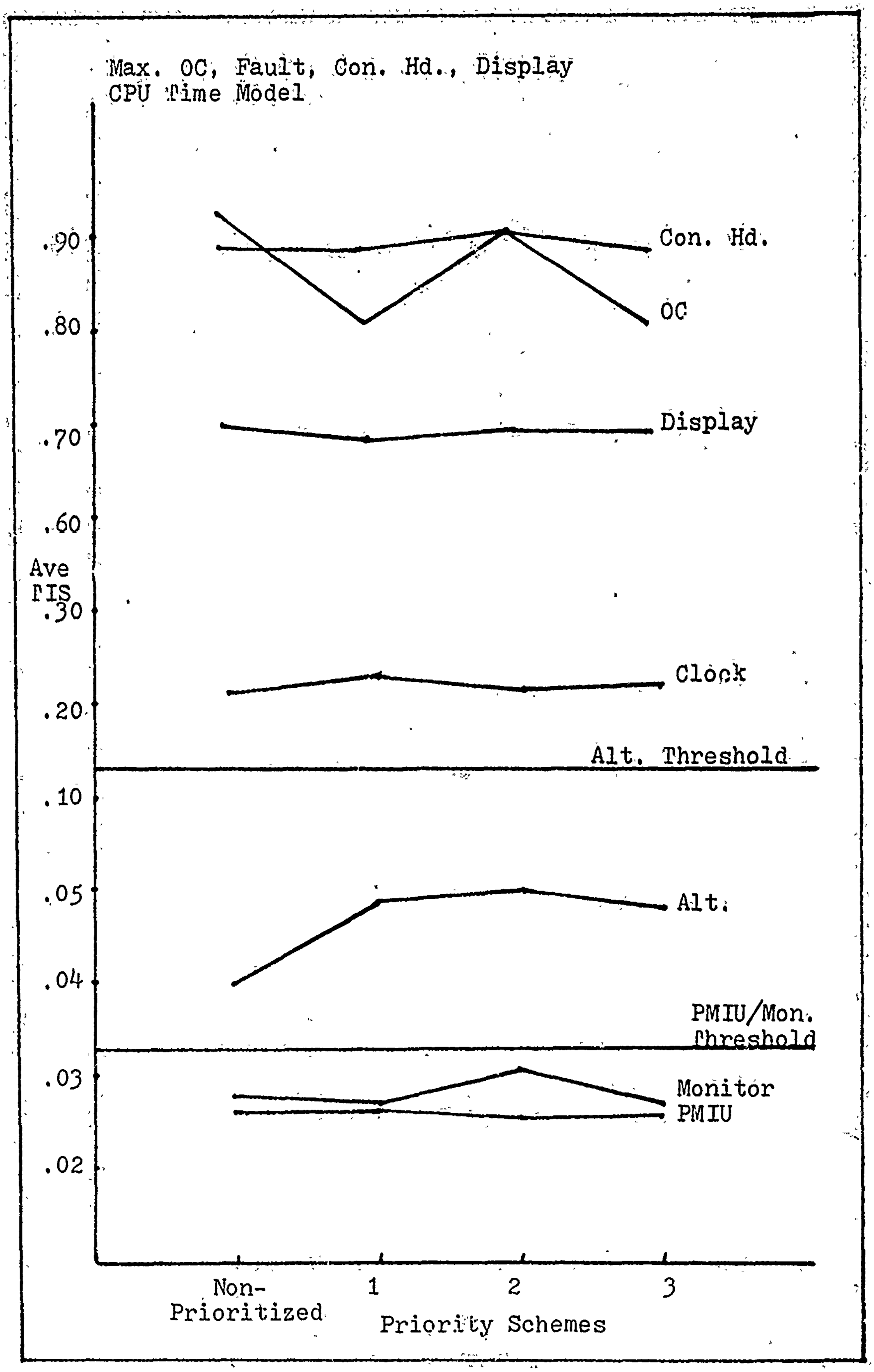
Results Of Experiment Four



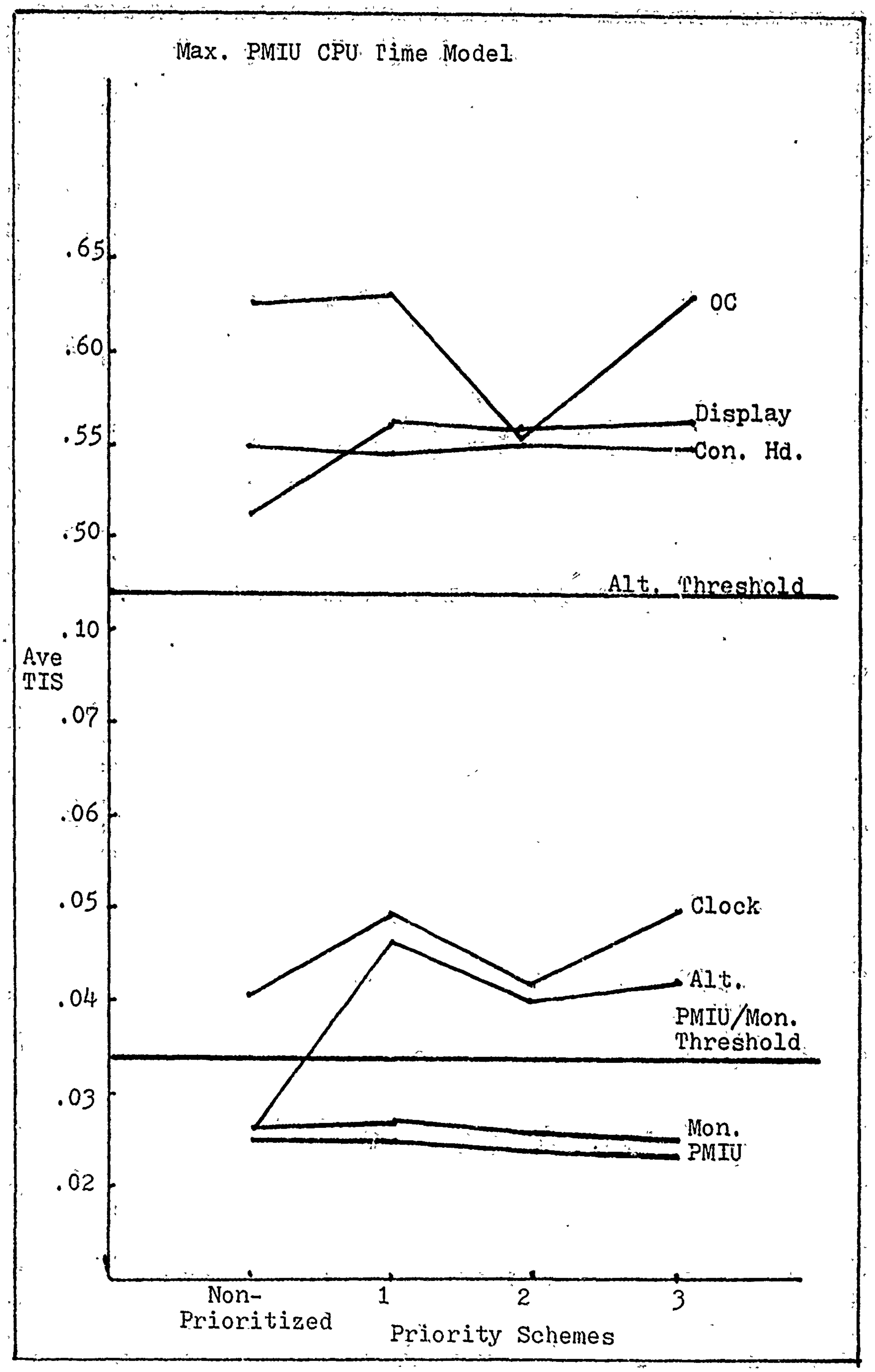
Results Of Experiment Four



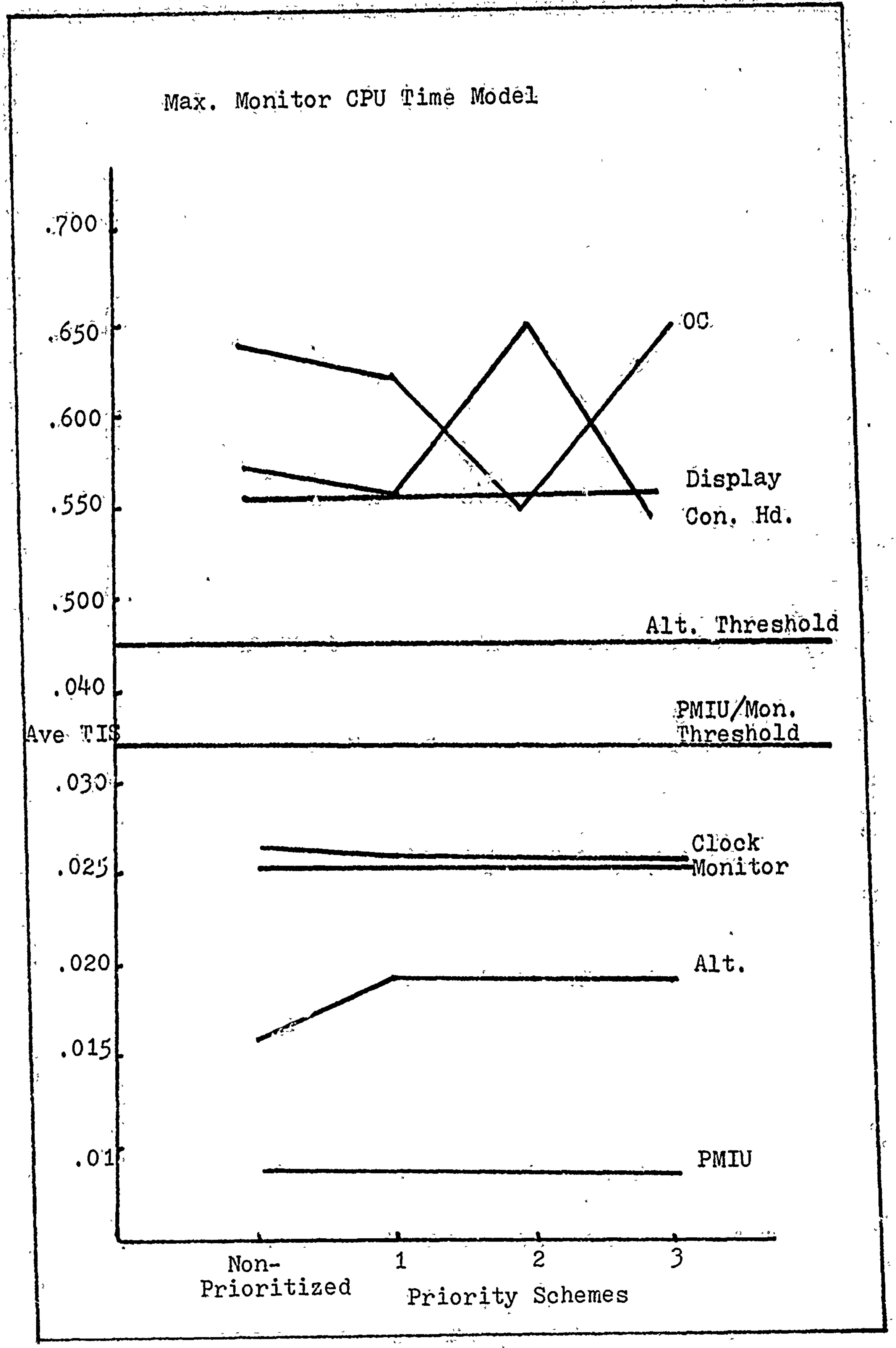
Results Of Experiment Four



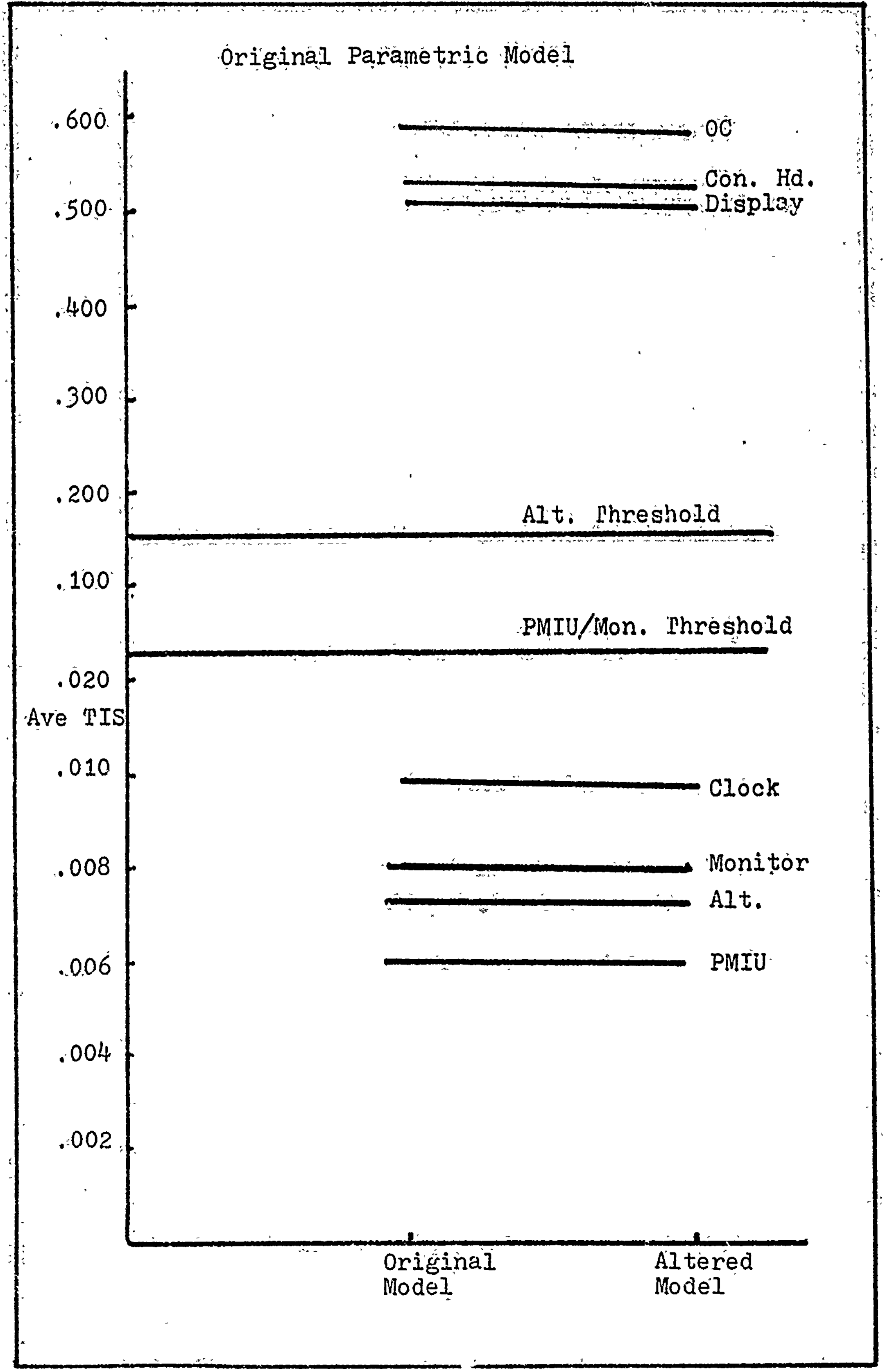
Results Of Experiment Four



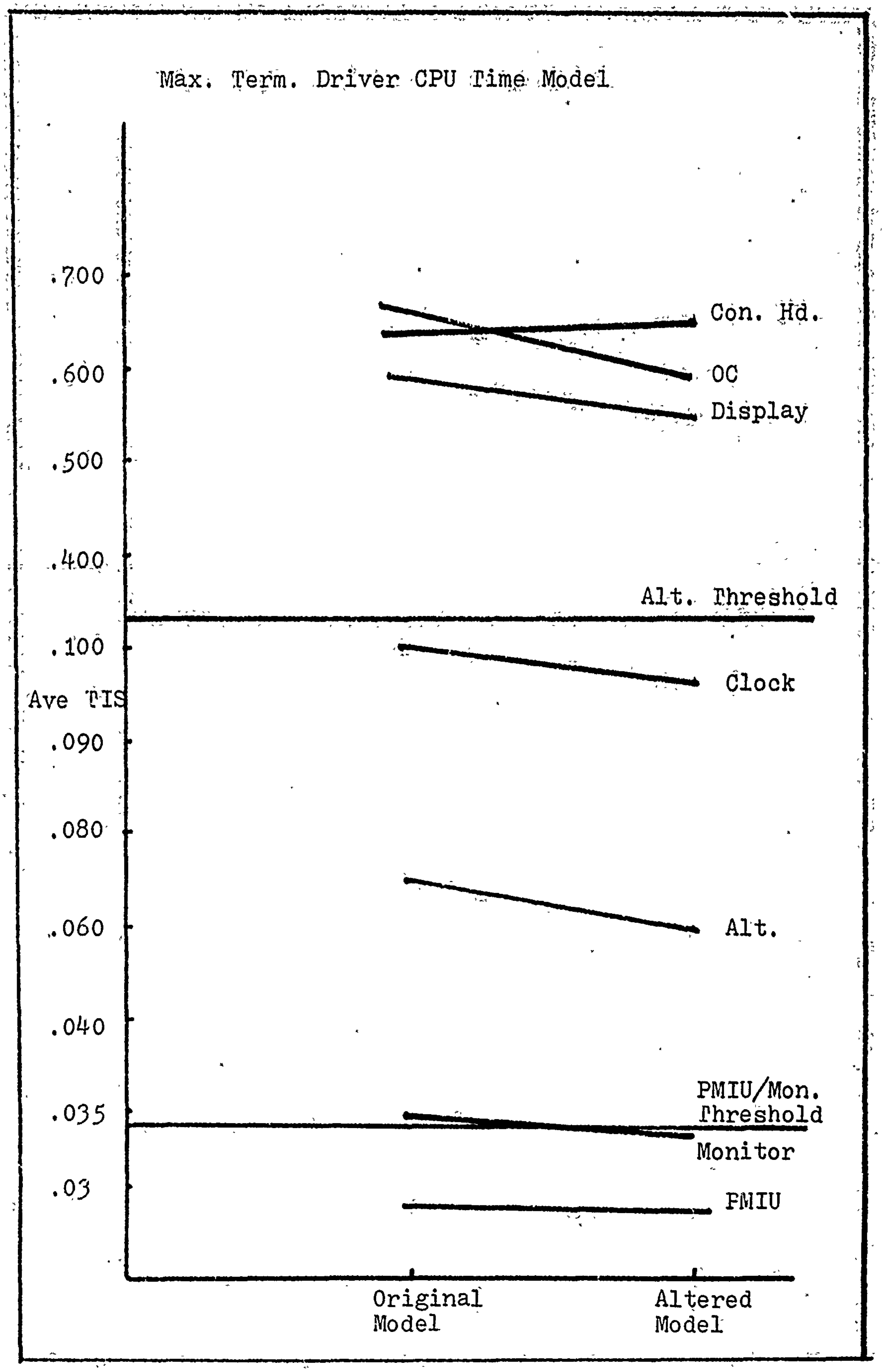
Results Of Experiment Four



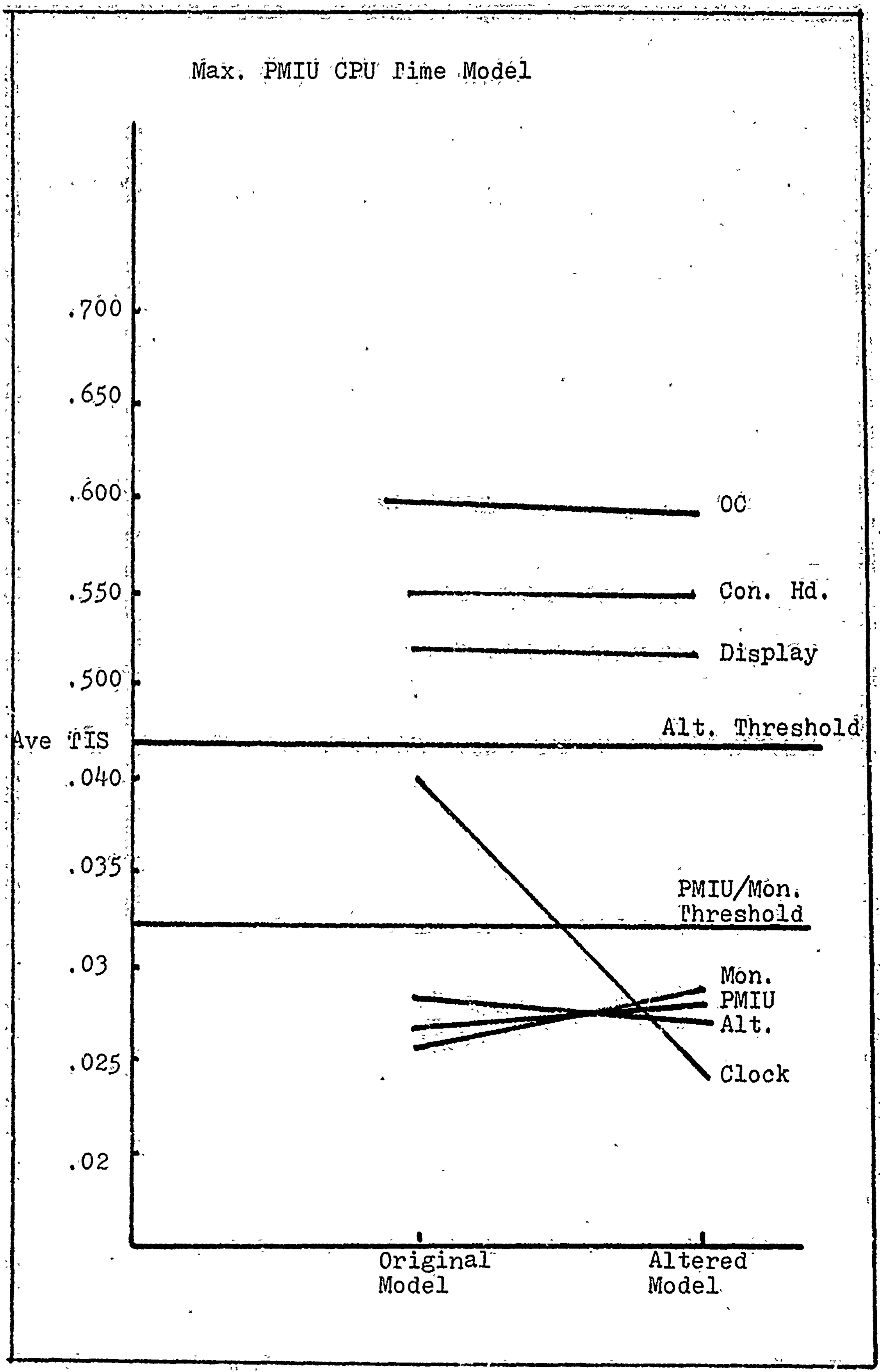
Results Of Experiment Five



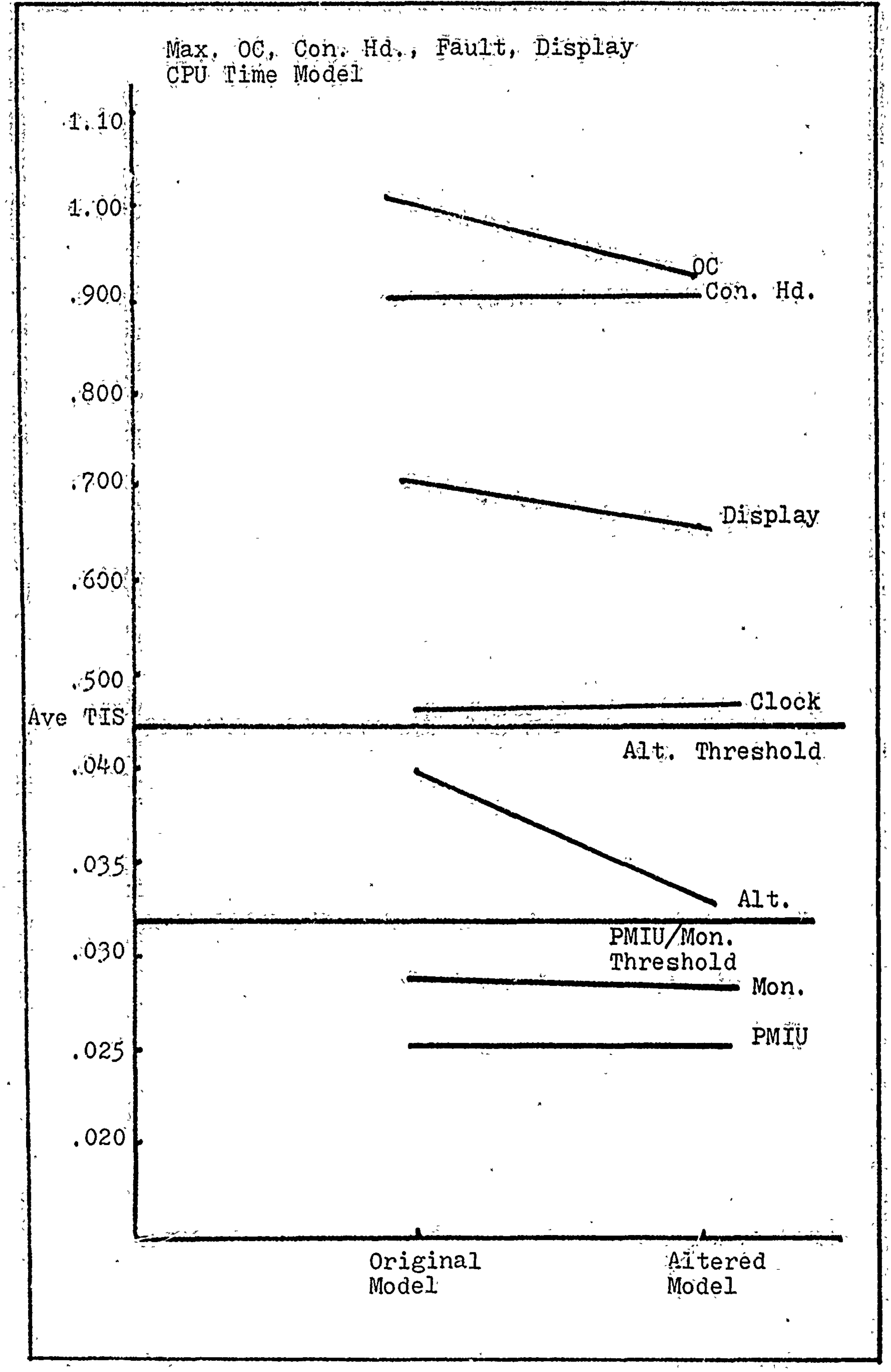
Results Of Experiment Five



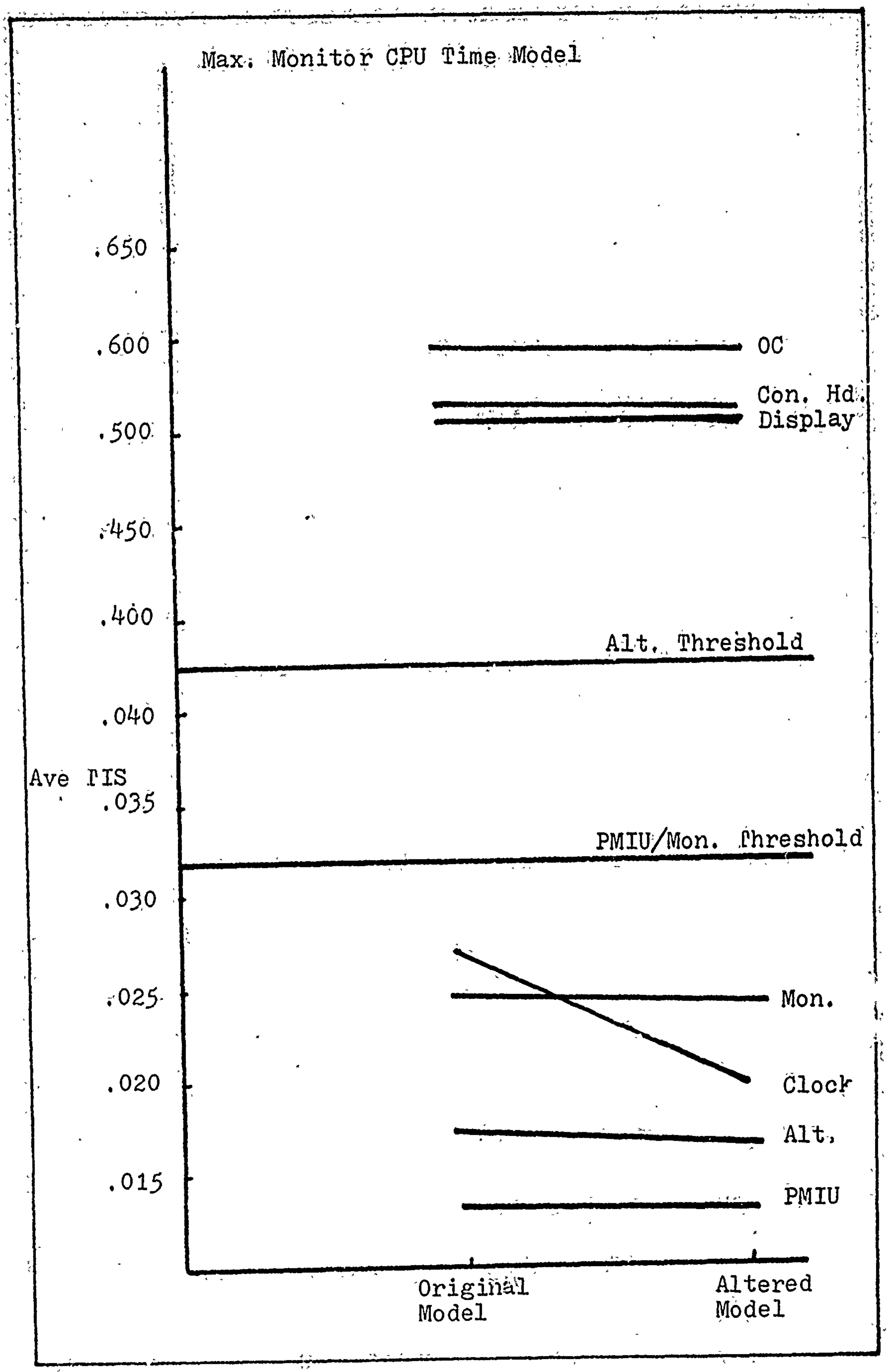
Results Of Experiment Five



Results Of Experiment Five



Results Of Experiment Five



Vita

Captain Jon G. Hanson was born on January 19, 1956 in Anchorage Alaska. In 1973 he graduated from Hampton High School in Hampton, Virginia. He attended the United States Air Force Academy from which he received a Bachelor of Science degree in 1977. Following graduation, he attended Communication Operations Officer School at Keesler AFB, Mississippi. Between April 1978 and April 1980 he was assigned to the 2192d Communications Squadron, Air Force Communications Command, at Loring AFB, Maine as the Combat Crew Communications Officer for the 42d Bombardment Wing. He entered the Air Force Institute of Technology in June 1980.

Permanent Address: 1941 Rainbow Dr.
Clearwater, Fla.
33515

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER AFIT/GCS/EE/81D-10	2. GOV'T ACCESSION NO. AD-A115 537	3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle) DESIGN AND IMPLEMENTATION OF USAF AVIONICS INTEGRATION SUPPORT FACILITIES	5. TYPE OF REPORT & PERIOD COVERED MS Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Jon G. Hanson, Captain, USAF	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, OH 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, OH. 45433	12. REPORT DATE December 1981 13. NUMBER OF PAGES 323
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

15 APR 1982

Dean for Research and
Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

Lynn E. Wolaver

Fre~~deric C. Lynch, Maj~~or, USAF
~~Director~~ of Public Affairs

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Avionics Integration Support Facility (AISF)
Standard AISF
Avionics Standardization
Integration Test Bed
Q-GERT

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See reverse

The design and implementation of avionic software support facilities known as Avionic Integration Support Facilities (AISFs) are investigated. A complete set of AISF functional requirements is presented in Data Flow Diagram format. This method of presentation facilitated the hierarchical development of the functional requirements. From these functional requirements an implementationmodel for such a facility is developed. The modeling vehicle used is Softech's Structured Analysis and Design Technique (SADT). The SADT model depicts inputs, outputs, processes, controls, and the mechanisms required to implement the AISF.

Following completion of the SADT model a specific implementation of a portion of an AISF, the Integration Test Bed (ITB), is examined. The ITB considered is an Avionics Modernization Hot Bench presently being developed by the Aeronautical Systems Division, System Engineering Avionics Facility (SEAFAC). A portion of the hot bench, the real time simulation software, is modeled and analyzed with the Queuing Graphical Evaluation and Review Technique (Q-GERT). Q-GERT is a modeling vehicle and computer analysis tool well suited for this purpose. In this way a predictive model of the real time simulation software is provided and the viability of the hot bench design verified.

A brief investigation of possible future AISF configurations is presented along with a discussion of present standardization efforts in the avionic arena and their impacts on avionic software support.